AFIT/GOR/ENS/98M-24

THE APPLICATION OF SEQUENTIAL CONVEX PROGRAMMING TO LARGE-SCALE STRUCTURAL OPTIMIZATION PROBLEMS

THESIS

Todd Allen Sriver
Captain, USAF

AFIT/GOR/ENS/98M-24

AFIT/GOR/ENS/98M-24

THE APPLICATION OF SEQUENTIAL CONVEX PROGRAMMING TO LARGE-SCALE STRUCTURAL OPTIMIZATION PROBLEMS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Operations Research

Todd Allen Sriver, B.S.

Captain, USAF

March, 1998

AFIT/GOR/ENS/98M-24

THE APPLICATION OF SEQUENTIAL CONVEX PROGRAMMING TO LARGE-SCALE STRUCTURAL OPTIMIZATION PROBLEMS

Todd Allen Sriver, B.S.

Captain, USAF

Approved:

[illegible] 16 Mar 98

Dr. James W. Chrissis
Thesis Advisor
Date

[illegible] 16 March 98

Maj. Edward A. Pohl
Reader
Date

Acknowledgements

I express my sincere gratitude to those individuals who helped make this experience so rewarding. I would like to thank, in particular, my thesis committee, whose guidance and support helped me focus clearly on my objectives. My advisor, Dr. James Chrissis, provided his expertise and encouragement throughout this journey in addition to adept technical editing recommendations. My reader, Major Edward Pohl, generously applied his particular expertise to the project, ensuring nothing was overlooked.

I am also deeply indebted to the people of the Design and Methods Development branch at the Air Force Research Laboratories. My sponsor, Dr. Vipperla Venkayya, was extremely accommodating to my needs, providing unlimited access to his busy staff and always finding a computer for me to work on. I thank Vicki Tischler for her expertise with ASTROS and the test problems. I thank Lt. Gerald Andersen and Dr. Ray Kolonay (formerly of the Laboratory) for their help in manipulating ASTROS so it would do what I wanted.

I also owe thanks to classmate Capt. Jonathon Clough, who answered countless of my LaTeX and MATLAB questions so that this document might look decent.

A special thanks goes to Dr. Christian Zillober of the University of Bayreuth, Germany. Without his software, this thesis would have taken a much different direction.

Most importantly, I thank my wife, Stephanie, my son, Tyler, and my daughter, Kayla. Their patience, understanding, and unwavering love have given this experience special meaning.

Todd Allen Sriver

Table of Contents

List of Figures

List of Tables

AFIT/GOR/ENS/98M-24

Abstract

Structural design problems are often modeled using finite element methods. Such models are often characterized by constraint functions that are not explicitly defined in terms of the design variables. These functions are typically evaluated through numerical finite element analysis (FEA). Optimizing large-scale structural design models requires computationally expensive FEAs to obtain function and gradient values. An optimization approach which uses the SCP sequential convex programming algorithm of Zillober, integrated as the optimizer in the Automated Structural Optimization System (ASTROS), is tested. The traditional approach forms an explicitly defined approximate subproblem at each design iteration that is solved using the method of modified feasible directions. In an alternative approach, the SCP subroutine is called to formulate and solve the approximate subproblem. The SCP method is an implementation of the Method of Moving Asymptotes algorithm with five different asymptote determination strategies. This study investigates the effect of different asymptote determination strategies and constraint retention strategies on computational efficiency. The approach is tested on three large-scale structural design models, including one with constraints from multiple disciplines. Results and comparisons to the traditional approach are given. The largest of the three models, which had 1527 design variables and 6124 constraints, was solved to optimality with ASTROS for the first time using a mathematical programming method. The structural weight of the resulting design is 9% lower than the previously recorded minimum weight.

THE APPLICATION OF SEQUENTIAL CONVEX PROGRAMMING TO LARGE-SCALE STRUCTURAL OPTIMIZATION PROBLEMS

I. Introduction

Engineering designs are often modeled using mathematical models with functions representing their performance characteristics and physical attributes. As the complexity of engineering designs increases, so too do the expectations for designs that minimize cost and maximize performance. Optimization theory provides a natural tool to help meet these sometimes conflicting objectives.

Engineering optimization problems involve the optimization of some design criterion subject to various constraints on the design parameters. When the design of a structure (a system of spars, trusses, beams, etc.) is considered, it is often called a *structural optimization* problem.

1.1 Structural Optimization

The classical structural optimization problem seeks the design vector $\mathbf{x} = (x_1, \ldots, x_n)$ that optimizes an objective function f (typically minimizing structural weight) such that behavioral and side constraints are met. Denoting the optimization problem as (P), it can be stated in the following mathematical form:

$$(\mathrm{P}) \quad \min \ f(\mathbf{x})$$

subject to

$$g_j(\mathbf{x}) \leq 0 , \quad j = 1, \ldots, m$$

$$\mathbf{x_l} \leq \mathbf{x} \leq \mathbf{x_u}$$

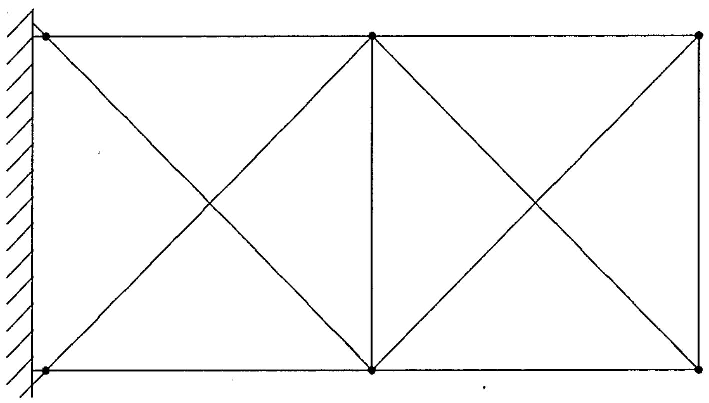


Figure 1.1 Ten-bar Truss

where g_j ($j = 1,\ldots,m$) are the behavioral constraints, m is the number of constraint inequalities, n is the number of design variables, and $\mathbf{x_l}$, $\mathbf{x_u}$ are the lower and upper bound vectors (side constraints), respectively, for the design variables. The functions f and g_j ($j = 1,\ldots,m$) may be assumed nonlinear in terms of the design variables $\mathbf{x}$.

Structures are often modeled using discrete finite elements. In a finite-element model, each element connects a set of grid points. An example of a simple finite-element model is the classic ten-bar truss, depicted in Figure 1.1. Each element of a finite-element model contributes one or more design variables to the optimization problem [46], typically representing sizing or shape parameters of structural components.

Behavioral constraints in structural models typically consist of restrictions on the structural response quantities. For mechanical structures, the most common constraints are placed on component stresses under static load but may also include displacement, frequency, and buckling constraints [44]. For large aerospace structures, the behavioral constraints may additionally include restrictions on response quantities from multiple disciplines. Examples include flutter speed, divergence speed, and lift-curve slope [1].

For finite-element models, the explicit forms of the constraint functions are generally not known [18]. As a consequence, for any given design the constraint responses must be evaluated numerically through a finite-element analysis (FEA). Many automated design tools have been developed to accommodate the computational requirements of finite-element analysis (*e.g.* see [10] and [21]). Typically, these tools can evaluate constraint responses and the constraint gradients, utilizing this information to improve the design via optimization methods.

Advancements in automated tools and higher costs of production motivate the goal of designing the *total system*, rather than just a collection of the individual components. This methodology impacts structural optimization applications because it may introduce constraints from disciplines other than structures into the mathematical model. For example, a typical aircraft design encompasses elements from structures, aerodynamics, controls, and propulsion [48]. The Automated Structural Optimization System (ASTROS) is a finite-element-based software package that is often used for the design and modification of aerospace structures. A key objective of ASTROS is to provide a design tool that can simultaneously design to constraints from multiple disciplines. Constraints considered by ASTROS include: stress-strain, displacement, modal frequency, aeroelastic effects (lift effectiveness and aileron effectiveness), and flutter response [27].

In addition to their multidisciplinary nature, most practical aerospace structures are large: in excess of 50 design variables and a similar number of constraints [48]. When the design model is large, optimization methods that act directly on the given problem to improve the design are computationally inefficient. For this reason, approximations that reduce the dimension of the design space are often used in FEA applications. A common approximation technique involves replacing the design problem with a series of lower-dimensional approximate subproblems that, when solved in succession, converge to the optimal solution of the larger problem. Figure 1.2 shows how this technique is incorporated into a typical automated optimization process for structural designs (adapted from [44]). Because the constraint responses and gradients can be retrieved from the finite-element analysis, the subproblems are readily formed using a first-order Taylor-series expansion about a trial design point. The optimization process is halted when an appropriate convergence criterion is satisfied.

1.2 Problem Statement

Even with the use of approximation concepts, structural applications can still be computationally burdensome, particularly for very large problems. The essential problem

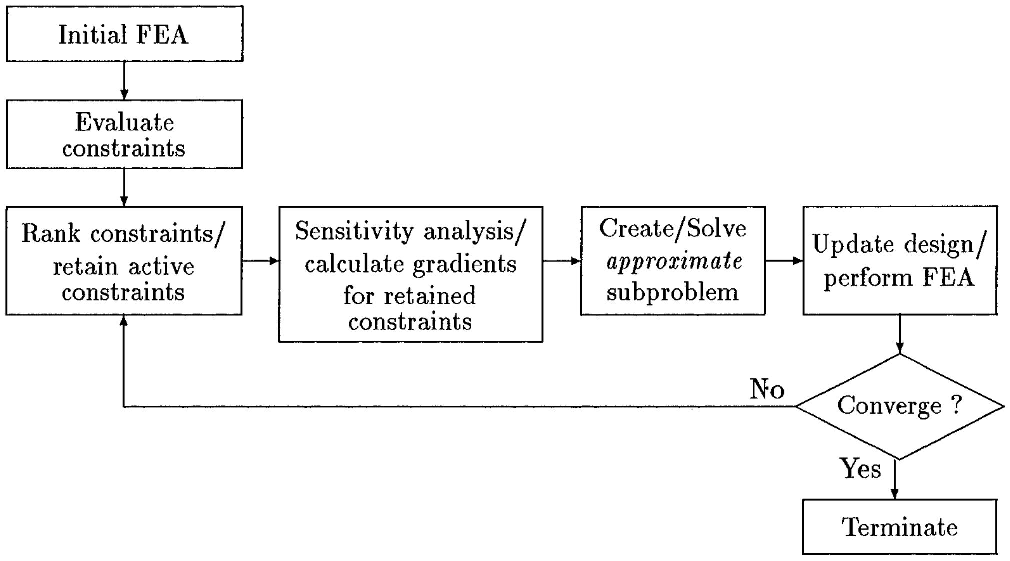

Initial FEA
Evaluate constraints
Rank constraints/ retain active constraints
Sensitivity analysis/ calculate gradients for retained constraints
Create/Solve *approximate* subproblem
Update design/ perform FEA
Converge ?
No
Yes
Terminate

Figure 1.2 FEA/Optimization Process

is that function and gradient calculations through finite-element analysis are expensive. This problem can be compounded if the optimization method is slow to converge, thereby leading to more design iterations and hence more FEAs. Clearly, to obtain optimal designs as efficiently as possible, it is necessary to:

- minimize the number of design iterations,
- minimize the number of gradient evaluations at each iteration, and
- employ an efficient optimizer.

The objective of this research is to seek efficient solutions to large-scale structural optimization problems using the ASTROS design tool as the test platform. In particular, an alternate optimization algorithm is tested. This algorithm belongs to the class of nonlinear programming algorithms known as *sequential convex programming.*

1.3 Sequential Convex Programming

The concept of solving a large nonlinear optimization problem with a sequence of approximate subproblems is well established. Two well-known methods that use this technique are sequential linear programming (SLP) and sequential quadratic programming (SQP). The SLP approach solves a sequence of subproblems that are linear approximations of the original problem. Similarly, SQP solves a sequence of approximated subproblems in which the objective function is quadratic and the constraint functions are linear.

Although not as well known, sequential convex programming (SCP) methods are similar in concept to SLP and SQP. SCP methods replace the implicit structural optimization problem with a sequence of convex explicit subproblems of simple algebraic form [18]. SCP methods use an efficient dual solution technique to solve the approximate subproblem, taking advantage of its convexity and separability. One such SCP method, CONLIN (for *CONvex LINearization*), forms an approximate subproblem using a first-order Taylor-series expansion about the current design point with respect to a mixed set of direct and reciprocal design variables [15]. The Method of Moving Asymptotes (MMA), another SCP method, generates a subproblem with the use of specialized parameters, called *moving asymptotes*, that are designed to stabilize and speed up convergence [38]. The MMA is considered a generalization of CONLIN. A third SCP method further stabilizes the MMA with the addition of a line search procedure [53].

1.4 Purpose and Approach

Sequential convex programming methods have attained success in solving structural optimization problems because they provide high quality approximations for structures under static loads with stress-strain constraints. However, current literature suggests that these methods have not been rigorously tested against very large problems, nor against problems of a multi-disciplinary nature. The hypothesis for this study is that an SCP optimizer, combined with an aggressive constraint retention strategy, can lead to efficient solutions for these problem types. The purpose of this research is to test this hypothesis against three large-scale structural design models.

For this study, Zillober's SCP method [51] was chosen for implementation into ASTROS to test against the structural design models. This algorithm is an implementation of the Method of Moving Asymptotes with an optional line search procedure that can be used to scale the redesign step size. Zillober's code is flexible in that it offers five different strategies for determining values for the asymptote parameters, including one scheme that approximates CONLIN.

To improve computational efficiency, this research investigates two areas:

1. An effective asymptote determination strategy, and
2. An efficient constraint retention strategy.

The asymptote determination strategy controls how the approximate subproblems are formed and directly correlates to the quality of the design point produced for the next design iteration. High quality design points can positively impact efficiency by minimizing the number of design iterations that are necessary. The constraint retention strategy determines the number of constraints that are retained for the subproblem. The fewer constraints that are retained, the fewer gradient calculations are required, saving computational expense.

The goal of this study is to learn if there exists any combination of strategies that can solve the test problems more efficiently than the traditional ASTROS approach. Results are compared to solutions obtained using the traditional approach in terms of computer processing (CPU) time, number of iterations required, number of gradients computed, and solution quality.

1.5 Overview

The next chapter describes background information and develops the methodology behind sequential convex programming methods as found in the literature. Chapter III describes the SCP method used and how it was integrated into the ASTROS environment. Chapter IV presents analysis of testing conducted against a small test problem for the purpose of obtaining good strategy choices, followed by comparative results of this implementation against the method currently used in ASTROS applied to the large structural design models described in Appendix A. Chapter V gives conclusions and recommendations for areas of further research.

II. Literature Review

This chapter summarizes the review of literature conducted in preparation for this study. Pertinent background information related to both structural optimization and sequential convex programming is described, thus laying the foundation for the remainder of the research.

2.1 Optimization Methods

Numerical methods for solving structural optimization problems are generally grouped into two categories: optimality criteria (OC), and mathematical programming (MP) methods [37]. OC methods use the Karush-Kuhn-Tucker (KKT) necessary conditions to formulate a recursive relationship for the design variables, iteratively converging to optimality. MP methods generate a sequence of improving design points using a computed search direction and step size at each design iteration.

2.1.1 Optimality Criteria Methods. Optimality criteria methods originated in response to the need for optimization of large practical structures [46]. As such, they existed in industry for some time with specialized application to the design of structures with a single constraint type (such as stress, displacement or frequency) [9]. More recently, Venkayya formulated a generalized optimality criteria approach for application to general mathematical functions that has enabled the application of OC methods to design optimization in a multi-disciplinary setting [46].

The two basic elements of OC methods are a *resizing* and a *scaling* algorithm [49, p. 154]. The resizing algorithm is derived directly from the optimality conditions and provides a simple recursive formula to update the design variables at each iteration. The purpose of the scaling algorithm is to locate the constraint boundary from anywhere in the n-dimensional space simply and economically. A scaling algorithm is important for constrained optimization problems because the optimum usually lies on the constraint boundary [49, p. 151].

A well-known resizing algorithm is derived in [46]. To develop optimality conditions, the Lagrangian function for problem (P) is first written as,

$$L(\mathbf{x},\lambda) = f(\mathbf{x}) + \sum_{j=1}^{m} \lambda_j g_j(\mathbf{x}) \tag{2.1}$$

where λ is the vector of Lagrange multipliers. The stationary condition of the KKT necessary conditions requires that the optimal point satisfy,

$$\frac{\partial f(\mathbf{x})}{\partial x_i} + \sum_{j=1}^{m} \lambda_j \frac{\partial g_j(x)}{\partial x_i} = 0, \quad i = 1,\ldots,n\ . \tag{2.2}$$

Equations (2.2) can be rewritten as

$$\sum_{j=1}^{m} e_{ij}\lambda_j = 1, \quad i = 1,\ldots,n \tag{2.3}$$

where

$$e_{ij} = -\frac{\partial g_j(\mathbf{x})/\partial x_i}{\partial f(\mathbf{x})/\partial x_i}\ . \tag{2.4}$$

Based on this condition, a simple resizing formula is formed as,

$$x_i^{\nu+1} = x_i^{\nu}\left[\sum_{j=1}^{m} e_{ij}\lambda_j\right]^{1/\alpha} \tag{2.5}$$

by multiplying both sides of Equation (2.3) by x_i^{α}. The parameter α is defined as the step size and ν is the iteration cycle.

The scaling algorithm forces the OC method to search along a constraint boundary. During the search, the solution vector is tested to determine whether its location lies on the constraint surface. If not, the solution is scaled according to the relation [49],

$$x_i^{new} = x_i^{old}\Lambda_{ij}\ , i = 1,\ldots,n \tag{2.6}$$

where Λ_{ij} are the scale factors. The scale factors are determined using rules for simple scaling or compound scaling. Their derivation is omitted here but can be found in [49].

2.1.2 Mathematical Programming Methods. Mathematical programming methods are intended to solve the general nonlinear programming problem. They are considered more robust than OC methods because of a wider applicability to many nonlinear problems. MP methods are based on the following iterative scheme:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)} \tag{2.7}$$

where k is the iteration number, $\mathbf{d}^{(k)}$ is the search direction at iteration k, and α_k is the step size along the search direction at iteration k.

Numerous methods implement numerical search techniques based on Equation (2.7), details of which can be found in standard engineering optimization or nonlinear programming textbooks (*e.g.* [3], [23], [32], [33], or [43]). A reason for the large number of methods is that no single method can solve efficiently all optimization problems [23, p. 29]. Only a general overview of the methods is given in this section.

MP methods are often distinguished based on the solution space that is used to solve the problem under consideration. *Primal* methods solve the problem in the solution space of the primal variables (*i.e.* the design variables) where *dual* methods solve the problem in the solution space of the dual variables. Some of the more well-known primal methods are the *constrained steepest descent method*, the *conjugate gradient method*, the *method of feasible directions*, the *gradient projection method*, the *generalized reduced gradient method*, *sequential linear programming*, and *sequential quadratic programming* ([3] and [32]).

Primal methods are much more common than dual methods but special conditions may exist which make a dual method a more efficient technique. For example, if the problem in question is convex, separable in terms of the design variables, and the number of design variables exceeds the number of active constraints, a dual method can solve the problem much more efficiently than a primal method [17, p. 510]. Sequential convex programming methods use dual methods of solution. A detailed discussion of dual solution methods is therefore covered in Section 2.4.

2.2 Approximation Concepts

Optimization of practical structural designs is encumbered by the large number of design variables and the implicit nature of the constraints—both of these characteristics adding considerable computational expense to finding good solutions. The quest for efficiency has led to innovative methods for dealing with the computational burden. Approximation concepts have evolved over the years and have been integrated into solution methods by replacing the problem under consideration with a sequence of relatively small, explicit, approximate subproblems that preserve the essential features of the original design problem [37, p. 31]. Three approximation techniques exist to meet this goal (from [36]):

1. Reducing the number of independent design variables by employing design variable linking.
2. Reducing the number of constraints to be evaluated by considering only the most critical constraints (*i.e.* violated, active, or near-active constraints).
3. Approximating the retained constraints by employing a Taylor-series expansion to explicitly approximate each constraint function.

2.2.1 Design variable linking. Also known as *basis reduction*, design variable linking fixes the relative size of some preselected group of members of a finite-element model so that *some* of the independent design variables control the size of *all* members [33]. This is accomplished by relating the original design variables $\mathbf{x}$ to the independent design variables $\mathbf{x}_R$ through a matrix of linking constants T,

$$\mathbf{x} = T \cdot \mathbf{x}_R \ . \tag{2.8}$$

The entries of matrix T contain predetermined ratios between the variables of $\mathbf{x}$ and $\mathbf{x}_R$.

2.2.2 Constraint set reduction. The concept of selecting only the most critical constraints can be discussed in two different contexts. First, when an MP algorithm is seeking a valid search direction, it may need to evaluate some constraint functions, depending on the method being used. It makes sense that only the constraint boundaries in the region of the current design point be considered when making this determination. As a result, the method may be implemented with a *potential constraint strategy* in which only the violated, active, and near-active constraints are considered at an iteration point [3, pp. 355–356].

Secondly, when a sequence of approximate subproblems is used to solve the larger main problem, an *active set strategy* may be used. In this methodology, the construction of the subproblem requires the gradient values for all constraints that are to be included in the subproblem. If some constraints are eliminated from this set (*i.e.* the inactive constraints), then the method is more efficient because the number of gradient evaluations is reduced [37, p. 33]. Since the approximation is only valid over a relatively small region (particularly for first-order approximations), it makes sense to exclude inactive constraints that are far from the current design point.

2.2.3 Function approximations. As mentioned previously, the constraint responses for finite-element models in structural optimization are typically implicit in terms of the design variables—the explicit nature of the functions are not known. When constructing subproblems to be solved sequentially, it is therefore necessary to approximate these functions explicitly. Most function approximations used in structural optimization are first-order because many finite-element software packages have the capability to compute the first derivative quantities but not second-order or higher-order derivatives.

The earliest first-order function approximations to be used were the linear approximations, obtained by expanding a Taylor series about a design point with respect to the *direct* variables (*i.e.* the design variables themselves). The linear approximation for a function $c(\mathbf{x})$ with respect to n direct variables x_i yields the following approximation,

$$c(\mathbf{x}) = c(\mathbf{x}^o) + \sum_{i=1}^{n} c_i^o (x_i - x_i^o) \tag{2.9}$$

where c_i^o denotes the first derivative of $c(\mathbf{x})$ with respect to variables x_i evaluated at $\mathbf{x}^o$. The early interest in approximating nonlinear problems in this manner was driven by the success of the simplex algorithm for linear programming models. This spawned research in sequential linear programming methods for structural optimization problems [4].

A drawback of the linear approximation is its inaccuracy, even for design points that are near the expansion point. Research was directed toward finding higher quality approximations without increasing the order of the expansion. As a result of studies on structures consisting of truss or plane stress elements, an approximation was developed that applies a first-order Taylor-series expansion with respect to *reciprocal* variables, $1/x_i$ [4]. The result for function $c(\mathbf{x})$ is,

$$c(\mathbf{x}) = c(\mathbf{x}^o) + \sum_{i=1}^{n} (x_i^o)^2 c_i^o \left(\frac{1}{x_i} - \frac{1}{x_i^o} \right) . \tag{2.10}$$

For structures in which the design variables are the cross-sectional areas of truss elements and thicknesses of plane-stress elements, Equation (2.10) yields an approximation for stress and displacement constraints that is exact for *statically determinate structures* [4, p. 54].

Continued research in function approximations resulted in the development of a class of approximations that yield convex functions. Since these approximations are the cornerstone for sequential convex programming methodology, they are discussed in detail in the following section.

2.3 Convex Approximation Methods

A function is said to be convex if a line segment joining two points on the function curve lies entirely above or on the curve [32, p. 876]. Convex approximations are formed by linearizing a function about an expansion point with respect to appropriately chosen *intermediate* variables. Fleury and Braibant were the first to propose a function approximation to exploit convexity [12]. Their CONLIN method generates a convex function by linearizing a given function $c(\mathbf{x})$ with respect to a *mixed* set of direct variables x_i and reciprocal variables $1/x_i$, depending on the sign of the first derivatives. This yields the following *convex* approximation,

$$c(\mathbf{x}) = c(\mathbf{x}^o) + \sum_{+} c_i^o (x_i - x_i^o) - \sum_{-} (x_i^o)^2 c_i^o \left(\frac{1}{x_i} - \frac{1}{x_i^o} \right) \tag{2.11}$$

where c_i^o denotes the first derivative of $c(\mathbf{x})$ with respect to design variable x_i evaluated at $\mathbf{x}^o$. The symbol $\sum_+(\sum_-)$ represents the summation over the terms for which c_i^o is positive (negative) [18]. An important characteristic of this approximation is that it is the most conservative approximation amongst all combinations of mixed direct/reciprocal variables [15]. As a consequence, the approximation is valid over a larger region in the solution space than an approximation using any other combination.

The Method of Moving Asymptotes (MMA), another first-order method proposed by Svanberg, linearizes $c(\mathbf{x})$ with respect to the intermediate variables $1/(x_i - L_i)$ and $1/(U_i - x_i)$, again depending on the sign of the derivatives. The approximation is of the form [38]:

$$c(\mathbf{x}) = \sum_+ \frac{c_i}{(U_i - x_i)} - \sum_- \frac{c_i}{(x_i - L_i)} - d^o \tag{2.12}$$

where

$$\begin{aligned} c_i &= (U_i - x_i^o)^2 \left(\frac{\partial c}{\partial x_i}\right)^o \quad \text{if } \left(\frac{\partial c}{\partial x_i}\right)^o > 0 \\ c_i &= -(x_i^o - L_i)^2 \left(\frac{\partial c}{\partial x_i}\right)^o \quad \text{if } \left(\frac{\partial c}{\partial x_i}\right)^o < 0. \end{aligned}$$

The constant d^o includes the zero-order terms of the expansion. The L_i and U_i parameters act as asymptotes that may "move" at each iteration, hence the name of the method. It can be shown that when $L_i = 0$ and $U_i = +\infty$, this is equivalent to CONLIN and when $L_i = -\infty$ and $U_i = +\infty$, this is equivalent to an SLP approach. The motivation for this approach is to further generalize the approximation so that greater flexibility exists for the types of constraints that can be handled.

An important characteristic of the first-order convex approximations is that they are *separable*. A function is said to be separable when it can be expressed as the sum of functions of the individual variables [43], *i.e.*, if the function $F(\mathbf{x})$ is separable, then:

$$F(\mathbf{x}) = f_1(x_1) + f_2(x_2) + \ldots + f_n(x_n) \ . \tag{2.13}$$

The significance of separability for convex nonlinear programming problems is discussed in the next section. It is introduced here to be discussed in context with second-order convex approximations.

Second-order approximations are not separable in general, but measures may be taken to ensure separability. A separable, second-order convex approximation was developed by Fleury [14], who proposed an extension to the MMA using second-order information to compute the asymptotes. Several improvements have been suggested in the literature to further generalize the method (*e.g.* see [7]). Fleury also introduced a "diagonal SQP" method that ensures separability of the approximation and makes the method suitable for larger problems [13]. Second-order methods are advantageous because of the high quality approximations that are obtained [24]. However, since second derivative information is required, they are not generally considered computationally feasible for large-scale finite-element applications.

2.4 Dual Solution Methods

As mentioned in Section 2.1.2, dual solution methods are efficient for optimization problems that are both convex and separable. An optimization problem is said to be *convex* if the Hessian matrices of the objective function and all the constraints are positive semi-definite [50]. The attractive property of a convex problem is the uniqueness of the optimal solution, which equals the minimum of the primal and the maximum of its dual. Therefore, if it is more efficient to solve the dual than the primal problem computational savings can be achieved.

Formulation of the dual problem is derived from the Lagrangian function of the convex problem. Recall the definition of the Lagrangian function for the primal problem (P) from Equation (2.1) and note that the vector of Lagrange multipliers λ are also the dual variables. Repeating Equation (2.2), the stationary condition of the KKT necessary conditions requires that the optimal point satisfy,

$$\frac{\partial f(\mathbf{x}^*)}{\partial x_i} + \sum_{j=1}^{m} \lambda_j^* \frac{\partial g_j(\mathbf{x}^*)}{\partial x_i} = 0, \quad i = 1, \ldots, n\,. \tag{2.14}$$

Separability of f and g_j $(j = 1, \ldots, m)$, and therefore L, implies from Equation (2.14) that there is a unique correspondence between the optimal $\mathbf{x}^*$ and λ^*. Hence, the design variables can be written explicitly in terms of the dual variables, which can be substituted back into the Lagrangian function to obtain the following dual problem [18]:

$$\max \ L(\lambda) \tag{2.15}$$

subject to

$$\lambda_j \geq 0, \ \ j = 1, \ldots, m \, . \tag{2.16}$$

The solution, λ^*, to this problem can then be substituted into the explicit expression $\mathbf{x}^* = \mathbf{x}(\lambda^*)$ to solve for the design variables.

The efficiency of the dual method can be explained by the following [37]:

- The algebraic structure of the approximate primal problem is exploited to obtain an explicit dual function in terms of the dual variables.
- Each explicit dual function is quasi-unconstrained (non-negativity constraints only).
- The dimensionality of the dual space is usually small because the number of dual variables corresponds to the number of active constraints, which is usually smaller than the number of design variables.

2.5 SLP and SQP

Section 2.2.3 referenced the early research work in applying SLP to structural optimization. Considering the generality and simplicity of the methodology, its appeal in an engineering design context is not surprising. The basic approach is to approximate the objective function and each critically selected constraint using a first-order Taylor-series approximation about the current solution vector. Solve this subproblem and use the solution vector as the expansion point in the next iteration. Repeat the process until an appropriate termination criterion is reached.

Despite its popularity, this method has been shown to have serious limitations. Already mentioned is the poor quality of approximations generated through conventional linearization (*i.e.* using direct variables). Another drawback of the method is that the sequence of approximate subproblems does not converge to a local minimum unless each solution to the linearly-approximated subproblems occurs at a vertex of the feasible domain. Otherwise, the optimization process may converge to a non-optimal vertex or oscillate indefinitely between two or more vertices [12]. It is also possible for the subproblem to produce a solution that is unbounded [3, p. 365]. These problems have been dealt with by the addition of artificial "move limits", which restrict the solution of the subproblem to a specified region (rectangular in 2-space). These move limits are gradually tightened, using an update formula, at each iteration so that the solution converges to the optimal solution of the original problem [12].

Addition of move limits to the SLP process poses complications because their selection as input parameters is critical to the success of the algorithm [3, p. 366]. Several trials may be necessary before proper values are chosen. The SQP methods were developed to alleviate the need for the choice of move limits. In this methodology, the subproblem is formulated with respect to the search direction. The constraints are still formed as linear approximations to the original problem. The objective function, however, is formed as a quadratic approximation to the Lagrangian function. Control parameters are added to the subproblem to ensure that the linearized constraints do not cut off the feasible space. The solution to the subproblem provides the direction of search. The step size is then determined by minimizing an appropriately chosen merit function. The process is then repeated, using the solution from the previous iterate as the expansion point until the termination criterion is satisfied [32, pp. 480–481]. Many implementations of this general method have been researched (see *e.g.* [41]). These methods are attractive because the subproblems can be solved by standard methods for solving quadratic programming problems and they have been shown to be robust; a drawback of the methods is that they can be sensitive to the control parameters [45].

2.6 SCP Methods

The basic procedure of SCP methods involves replacing the original problem with a series of convex and separable subproblems using one of the methods described in Section 2.3 to approximate the objective function and each active constraint. At each iterate, the dual of the subproblem is formed and solved in terms of the dual variables (Lagrange multipliers). The primal variables of the subproblem are then determined using the explicit relationship between the primal and dual variables. The approximation expansion point at each iteration is the solution of the subproblem at the previous iteration. This process is repeated until the termination criterion is satisfied [16].

SCP methods have an advantage over SLP that results from the convex curvature in the constraint function approximations. The convexity implies that the solution is unique and move limits are therefore not required. SCP can have advantages over SQP methods in terms of efficiency. By their nature, SQP methods are not separable, which implies that the dual problem cannot be formulated explicitly. Therefore, if the dual space is of lower dimension than the primal space (*i.e.* less active constraints than design variables), the SCP methods may be more efficient due to their dual solution method.

The SCP methods presented in this section are the CONLIN method, the Method of Moving Asymptotes (MMA), and the MMA augmented with a line search. The construction of a convex subproblem for the CONLIN method and the MMA is demonstrated for the following simple nonlinear programming problem, taken from [47]:

$$\begin{array}{lrl} \text{(P1)} & \min & f(\mathbf{x}) = 10x_1 + x_2 \\ & \text{subject to} & g_1(\mathbf{x}) = -2x_1 + x_2 + 1 \leq 0 \\ & & g_2(\mathbf{x}) = -x_1 + 2x_2 - 1 \leq 0 \\ & & g_3(\mathbf{x}) = x_1^2 - 2x_1 - 2x_2 + 1 \leq 0 \end{array}$$

using the solution vector $\mathbf{x}^o = (1,1)^T$ as the expansion point. Additionally, the dual solution method is illustrated for the CONLIN method. Note, that the optimal solution to (P1) is $\mathbf{x} = (0.55, 0.10)^T$ and $f = 5.61$.

2.6.1 CONLIN.

The CONLIN method uses the convex approximation of Equation (2.11) for the objective function and each active constraint to construct the subproblem. An attractive feature of the method is that it chooses the appropriate combination of mixed variables to use in the approximation based on the sign of the derivatives of each function with respect to each variable [12]. The only input required are the function values and gradient values for the objective function and the same for the active constraints—no control parameters are necessary.

The CONLIN method is now illustrated for problem (P1). The partial derivative quantities evaluated at $\mathbf{x}^o = (1,1)^T$ are required:

$$\begin{array}{ll}
\frac{\partial f}{\partial x_1} = 10 & \frac{\partial f}{\partial x_2} = 1 \\
\frac{\partial g_1}{\partial x_1} = -2 & \frac{\partial g_1}{\partial x_2} = 1 \\
\frac{\partial g_2}{\partial x_1} = -1 & \frac{\partial g_2}{\partial x_2} = 2 \\
\frac{\partial g_3}{\partial x_1} = 0 & \frac{\partial g_3}{\partial x_2} = -1 \ .
\end{array}$$

Using these quantities and the approximation technique of Equation (2.11), the following subproblem is generated:

$$\begin{array}{lrl}
\text{(SP1)} & \min & 10x_1 + x_2 \\
 & \text{subject to} & \frac{2}{x_1} + x_2 \leq 3 \\
 & & \frac{1}{x_1} + 2x_2 \leq 3 \\
 & & \frac{2}{x_2} \leq 4 \ .
\end{array}$$

To solve problem (SP1) using the dual method, state the Lagrangian function as,

$$L(\mathbf{x}, \lambda) = 10x_1 + x_2 + \lambda_1(\frac{2}{x_1} + x_2 - 3) + \lambda_2(\frac{1}{x_1} + 2x_2 - 3) + \lambda_3(\frac{2}{x_2} - 4) \ . \tag{2.17}$$

Differentiating with respect to x_1 and x_2 and setting equal to zero yields,

$$10 - \frac{2}{x_1^2}\lambda_1 - \frac{1}{x_1^2}\lambda_2 = 0 \tag{2.18}$$

$$1 + \lambda_1 + 2\lambda_2 - \frac{2}{x_2^2}\lambda_3 = 0. \tag{2.19}$$

Solve Equations (2.18) and (2.19) for x_1 and x_2 in terms of the dual variables,

$$x_1 = \sqrt{\frac{2\lambda_1 + \lambda_2}{10}} \tag{2.20}$$

$$x_2 = \sqrt{\frac{2\lambda_3}{1+\lambda_1+2\lambda_2}} \ , \tag{2.21}$$

and substitute these expressions back into the Lagrangian function to construct the dual problem:

$$\begin{array}{rcl}
\max \quad L(\lambda) & = & 10\sqrt{\frac{2\lambda_1+\lambda_2}{10}} + \sqrt{\frac{2\lambda_3}{1+\lambda_1+2\lambda_2}} \\
& & + \lambda_1 \left(2\sqrt{\frac{10}{2\lambda 1+\lambda_2}} + \sqrt{\frac{2\lambda_3}{1+\lambda_1+2\lambda_2}} - 3\right) \\
& & + \lambda_2 \left(\sqrt{\frac{10}{2\lambda_1+\lambda_2}} + 2\sqrt{\frac{2\lambda_3}{1+\lambda_1+2\lambda_2}} - 3\right) \\
& & + \lambda_3 \left(2\sqrt{\frac{1+\lambda_1+2\lambda_2}{2\lambda_3}} - 4\right) \\
\text{subject to} \quad \lambda_1, \lambda_2, \lambda_3 & \geq & 0 \ .
\end{array}$$

The optimal to the dual problem is $\lambda = (3.2, 0, 0.525)^T$. When substituted into Equations (2.20) and (2.21), the solution $\mathbf{x}^{(1)} = (0.8, 0.5)^T$ is reached. Upon repetition, the following sequence of points are generated: $\mathbf{x}^{(2)} = (0.66, 0.26)^T$, $\mathbf{x}^{(3)} = (0.59, 0.16)^T$, $\mathbf{x}^{(4)} = (0.56, 0.11)^T$, and $\mathbf{x}^{(5)} = (0.55, 0.10)^T$.

2.6.2 Method of Moving Asymptotes. Svanberg's Method of Moving Asymptotes approximates the objective function and the constraint functions using Equation (2.12). The obvious distinction between this method and CONLIN are the use of the asymptotes L_i and U_i, which may be different for each variable and may change at each iteration. The choice of these parameters is a key to the method's performance.

In choosing of L_i and U_i at iteration k, the following inequality must hold:

$$L_i^k < x_i^k < U_i^k, \quad i = 1, \ldots, n \tag{2.22}$$

where $\mathbf{x}^k = (x_1^k, x_2^k, \ldots, x_n^k)$ is the current design point. The closer L_i and U_i are to x_i^k, the more curvature is introduced into the approximating function, and the more conservative the approximated subproblem. Asymptotes chosen further away from x_i^k result in an approximation approaching linearity. The flexibility of MMA is fully exploited by allowing the asymptotes to change at each iteration based on the following rules [38]:

- If the convergence process tends to oscillate, it may be stabilized by moving the asymptotes closer to the current iteration point.
- If the convergence process is slow and monotone, it may be relaxed by moving the asymptotes away from the current iteration point.

Many implementations of these rules are possible. In his original work, Svanberg suggested the following strategy [38]. Here, s is a positive number less than one and $\underline{x}_i(\overline{x}_i)$ denote the lower (upper) bound on design variable x_i.

For $k = 0$ and $k = 1$,

$$L_i^k = x_i^{(k)} - (\overline{x}_i - \underline{x}_i) \quad \text{and} \quad U_i^k = x_i^{(k)} + (\overline{x}_i - \underline{x}_i) \,.$$

For $k \geq 2$,

(a) If $\text{sign}(x_i^{(k)} - x_i^{(k-1)}) \neq \text{sign}(x_i^{(k-1)} - x_i^{(k-2)})$,

$$\begin{aligned} L_i^k &= x_i^{(k)} - s(x_i^{(k-1)} - L_i^{k-1}) \\ U_i^k &= x_i^{(k)} + s(U_i^{k-1} - x_i^{(k-1)}) \,. \end{aligned}$$

(b) If $\text{sign}(x_i^{(k)} - x_i^{(k-1)}) = \text{sign}(x_i^{(k-1)} - x_i^{(k-2)})$,

$$\begin{aligned} L_i^k &= x_i^{(k)} - (x_i^{(k-1)} - L_i^{k-1})/s \\ U_i^k &= x_i^{(k)} + (U_i^{k-1} - x_i^{(k-1)})/s \,. \end{aligned}$$

Additional examples for parameter choice strategies are discussed in Section 3.4.

To demonstrate the method, a subproblem is generated here for problem (P1) using parameter choices $L_i^o = -2$, $(i = 1, 2)$ and $U_i^o = 2$, $(i = 1, 2)$. The required derivative information is the same as for the CONLIN example. Using Equation (2.12) to approximate all four functions, an approximate subproblem is obtained as,

$$\begin{array}{lrl} \text{(SP2)} & \min & \frac{10}{2-x_1} + \frac{1}{2-x_2} \\ & \text{subject to} & \frac{18}{x_1+2} + \frac{1}{2-x_2} \leq 7 \\ & & \frac{9}{x_1+2} + \frac{2}{2-x_2} \leq 5 \\ & & \frac{18}{x_2+2} \leq 8\ . \end{array}$$

The solution to this subproblem is $\mathbf{x}^{(1)} = (0.8, 0.25)^T$. After only two more iterations the method converges to two decimal places using the following parameter choices:

$$L_i^1 = -2,\ (i = 1,2), \quad \text{and} \quad U_i^1 = 2,\ (i = 1,2), \quad \text{at iteration 2}$$
$$(L_1^2, U_1^2) = (-3.43, 2.28) \quad \text{and} \quad (L_2^2, U_2^2) = (-3.13, 2.58), \quad \text{at iteration 3.}$$

The iterates produce the solution vectors $\mathbf{x}^{(2)} = (0.57, 0.08)^T$ and $\mathbf{x}^{(3)} = (0.55, 0.10)^T$.

2.6.3 MMA with a Line Search Procedure. Numerical experience with the SCP methods presented thus far has discovered one serious drawback: the methods are not guaranteed to converge [50]. For inappropriately chosen initial solution vectors, the solutions may diverge or oscillate indefinitely between infeasible designs [38]. In response, Zillober stabilized the MMA with the addition of a line search procedure [53]. The motivation for the method was to guarantee convergence from an arbitrary starting point.

Zillober's method, the Sequential Convex Programming (SCP) method to denote its similarity to SQP, computes a line search with respect to a merit function after the subproblem produces the search direction. The merit function, in this case, is an augmented Lagrangian function defined as,

$$\Phi(\mathbf{x}, \lambda) = f(\mathbf{x}) + \sum_{j=1}^{m} \begin{cases} \lambda_j g_j(\mathbf{x}) + \frac{r}{2} g_j^2(\mathbf{x}), & \text{if } g_j(\mathbf{x}) \geq -\frac{\lambda_j}{r} \\ \frac{\lambda_j^2}{2r}, & \text{otherwise} \end{cases} \tag{2.23}$$

where r is the penalty parameter. The penalty parameter must be sufficiently large and controls the degree of penalization when leaving the feasible region. This function is also used by Schittkowski in an SQP method [35].

Zillober theoretically proved global convergence of the method. The numerical results were somewhat mixed. For some test problems the MMA and SCP produced the same iteration sequence, for others the sequences differed but both reached optimality, and for the remaining problems MMA diverged but SCP converged to optimality [53].

2.7 Comparative Study Results

Published studies that compare the performance of SCP methods to other well-known optimization methods are rare. The lone exception is the comprehensive study of Schittkowski, Zillober, and Zotemantel [34]. Schittkowski et. al. compare the performance of eleven algorithms within the MBB-LAGRANGE structural optimization system on 79 test problems having up to 144 design variables and 1020 constraints. CONLIN, MMA, and SCP are all evaluated in the study. The study suggests that, for the SCP methods, algorithm reliability is only reasonably achieved with the use of safeguards (*i.e.* moving asymptotes or line search). The efficiency of the SCP methods compared favorably to the other methods. This is not surprising since most of the test problems were composed of primarily stress constraints. The methods performed surprisingly well, however, when mixed constraints were used. The authors theorize that the stress constraints dominate in these cases. The results also showed the SCP method to be only slightly more reliable than the MMA, a consequence of the relative proximity of the initial designs to the optimal solutions.

III. Approach

3.1 Overview

This chapter describes the details of the research conducted. Zillober's code, SCP, was chosen for testing because of its flexibility in choosing the asymptote parameters. Additionally, the algorithm is implemented via a FORTRAN subroutine with a special "reverse communication" logic which makes it easy to integrate into the ASTROS framework. A driver program was needed for the algorithm to interface with ASTROS. In this chapter, the current and proposed redesign step of the ASTROS optimization loop are contrasted, the SCP algorithm is described in greater detail, and an outline is given on how SCP was tested against large, multi-disciplinary structural models within ASTROS.

3.2 ASTROS Optimization Loop

ASTROS is an automated design and analysis tool that was developed to assist in the preliminary design of aerospace structures [30, p. 1]. The multi-disciplinary nature of these structures is accounted for in ASTROS using a modular approach; different ASTROS modules, which perform the necessary finite-element computations, represent the various disciplines of the structure under consideration. The ASTROS optimization procedure is driven by a sequence of calls to the modules, which feed the ASTROS optimization loop the data it needs to optimize the design of the structure. The standard sequence is written in the MAPOL (MAtrix Analysis Problem Oriented Language) programming language, a high level language that was designed to support the large-scale matrix operations typically encountered in engineering analysis [28, p. 475]. The user defines the structural design model to be optimized and tailors the standard MAPOL sequence to suit specific needs through a data input file.

ASTROS optimizes a given structural design according to the schematic presented in Figure 3.1 [30, p. 25]. The optimization is divided into three distinct phases: the *analysis* phase, the *sensitivity* phase, and the *optimization* phase. Of particular importance here is the optimization phase; this is where a redesign is performed to produce a new design point in the optimization process.

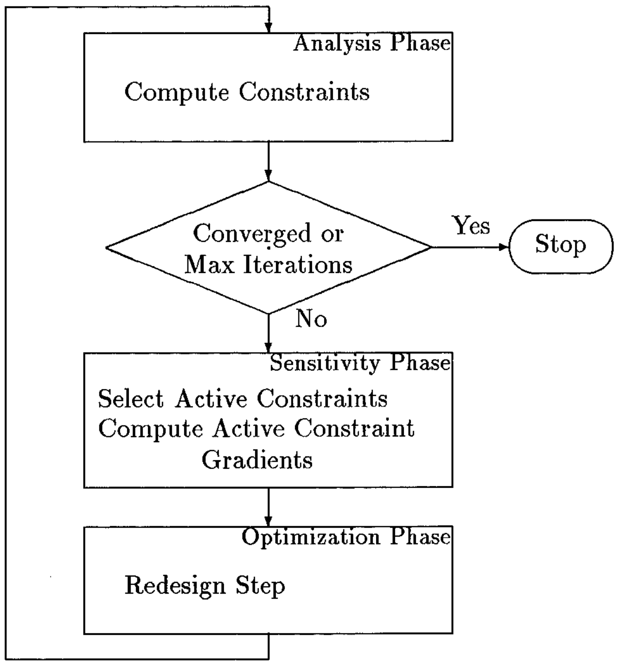


Figure 3.1 ASTROS Optimization Loop

The ASTROS optimization process uses all of the approximation techniques discussed in Section 2.2 to improve computational efficiency. Three design-variable linking options are available [30, pp. 28–29]:

- Unique linking–the global variables are the same as the local variables.
- Physical linking–one global variable uniquely specifies a number of local variables.
- Function linking–a local variable is the weighted sum of several global variables.

ASTROS also employs a constraint retention strategy, retaining only the active constraints during the sensitivity phase of Figure 3.1 for use in the redesign step. Active constraints are defined by the following criteria [30, p. 176]:

- All constraints with a value greater than a specified value, ε, are retained.
- The most active (**NRFAC** $\times$ *ndv*) are always retained, where **NRFAC** is a user-specified parameter and *ndv* is the number of global design variables.

Default values of **NRFAC** $= 3.0$ and $\varepsilon = -0.10$ are specified in the standard MAPOL sequence but may be tailored in the data input file, providing the user with significant control over how many constraints are included in the redesign. Computational savings can be gained because the sensitivities are calculated for only the active constraints.

In the optimization phase, the design module is called from the standard MAPOL sequence. This module constructs an approximate subproblem, using only the active constraints selected in the previous phase. This subproblem linearizes the objective function and each of the constraints with respect to direct or reciprocal variables using a first-order Taylor-series expansion [30, pp. 177–178]. The module then performs the redesign task by calling the μ-DOT optimization algorithm, which solves the approximate subproblem using the *method of modified feasible directions.* This method combines aspects of the method of feasible directions and the generalized reduced gradient method [42].

3.3 Alternative Redesign Step

For this study the SCP algorithm was used to construct and solve the approximate subproblem during each iteration of the optimization loop. This was accomplished by modifying the optimization phase of the ASTROS optimization loop in Figure 3.1 so that during the redesign step the SCP algorithm was called instead of the μ-DOT algorithm. A new design module (denoted as DESIGN2) was written to serve as a driver program for SCP and provide the interface between ASTROS and SCP. When using this implementation, the ASTROS standard sequence is altered so that DESIGN2 is called in lieu of the standard ASTROS design module. Details of the SCP integration into ASTROS are deferred until Section 3.5 so that a more thorough description of the SCP algorithm may be given.

3.4 The SCP Optimizer

Zillober implemented the SCP method as a FORTRAN subroutine [52]. The modular structure of the implementation is depicted in Figure 3.2 (from [51]). The user must provide the main program as well as the NLFUNC (function call) and NLGRAD (gradient call) subroutines. As noted in the figure, use of the line search (LINSEA) subroutine is optional. The user selects a method, either MMA, or MMA with line search, as an input to SCP to determine if the line search is performed.

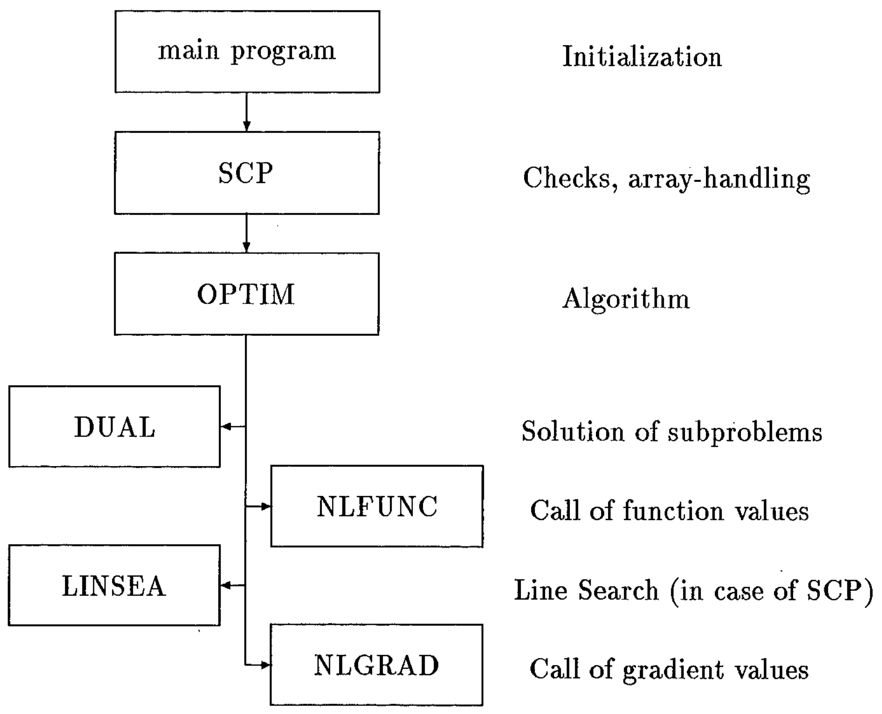


Figure 3.2 SCP Implementation

3.4.1 Algorithm. The algorithm, as presented in [51] is stated here. Recall that Φ is the augmented Langrangian function used in the line search, r is the penalty parameter, and ϵ is some positive number.

Step 0 : Choose $\mathbf{x}^o \in \mathbf{X}$, $\lambda^o \geq 0$, $0 < c < 1$, (*e.g.* 0.001), $0 < \psi < 1$ (*e.g.* 0.5), $r > 0$ (*e.g.* 1), let $k = 0$.

Step 1 : Compute $f(\mathbf{x}^k)$, $\nabla f(\mathbf{x}^k)$, $g_j(\mathbf{x}^k)$, $\nabla g_j(\mathbf{x}^k)$, $j = 1, \ldots, m$.

Step 2 : Compute L_i^k and U_i^k $(i = 1 \ldots n)$ by some scheme; define approximating functions $\tilde{f}(\mathbf{x})$, $\tilde{g}_j(\mathbf{x})$, $j = 1, \ldots, m$.

Step 3 : Solve subproblem; let $(\mathbf{y}^{k+1}, \nu^{k+1})^T$ be the solution, where ν^{k+1} denotes the corresponding vector of Lagrange multipliers.

Step 4 : If $\mathbf{y}^{k+1} = \mathbf{x}^k$ stop; $(\mathbf{x}^k, \lambda^k)^T$ is the solution.

Step 5 : Let $\mathbf{s}^k = (\mathbf{x}^k - \mathbf{y}^{k+1}, \lambda^k - \nu^{k+1})^T$, $\delta^k = \|\mathbf{y}^{k+1} - \mathbf{x}^k\|$,

$$\eta^k = \frac{1}{2}\min\left\{\min_{i=1\ldots n}\left\{2\epsilon\frac{(U_i^k - x_i^k)^2}{(U_i^k - L_i^k)^3}\right\}, \min_{i=1\ldots n}\left\{2\epsilon\frac{(x_i^k - L_i^k)^2}{(U_i^k - L_i^k)^3}\right\}\right\} (> 0) .$$

Step 6 : Compute $\Phi(\mathbf{x}^k, \lambda^k)$, $\nabla\Phi(\mathbf{x}^k, \lambda^k)$, $\nabla\Phi(\mathbf{x}^k, \lambda^k)^T\mathbf{s}^k$.

Step 7 : If $\nabla\Phi(\mathbf{x}^k, \lambda^k)^T\mathbf{s}^k < \eta^k(\delta^k)^2/4$, let $r = 10r$ and go to step 6; otherwise compute the smallest j such that

$$\Phi_r[(\mathbf{x}^k, \lambda^k)^T - \psi^j\mathbf{x}^k] \leq \Phi_r(\mathbf{x}^k, \lambda^k) - c\psi^j\nabla\Phi_r(\mathbf{x}^k, \lambda^k)^T\mathbf{s}^k; \text{ let } \sigma^k = \psi^j .$$

Step 8 : Let $(\mathbf{x}^{k+1}, \lambda^{k+1})^T = (\mathbf{x}^k, \lambda^k)^T - \sigma^k\mathbf{s}^k$, $k = k+1$; go to step 1.

The line search is carried out in step 5 through step 8 of the algorithm.

Three user-specified parameters, which are key to performance of the algorithm, are listed in Table 3.1. The first parameter, METHOD, has already been discussed. This determines whether the pure Method of Moving Asymptotes or the MMA with a line search is used. The STRAT parameter is probably the most important, as it determines how the asymptote parameters (L_i and U_i) are computed in step 2 of the algorithm, and thus is critical in determining the structure of the resulting subproblem. These strategies are covered in detail in the following subsection. The ACTRES parameter is the only means the SCP algorithm has for controlling the number of constraints that are included in the subproblem. Choice of this parameter impacts the computational efficiency of the algorithm because gradients are computed for only those constraints considered active.

3.4.2 Asymptote Determination Strategies. The five strategies implemented by Zillober are defined in this section. Note that strategy N requires the use of supporting parameters γ, $t1$, and $t2$, and strategy P requires the use of supporting parameter $t1$. The supporting parameters are restricted to the following ranges,

$$0.01 \leq \gamma \leq 1$$

Parameter	Purpose	Choices
METHOD	Determines which optimization method is used	'S' - MMA 'A' - MMA with line search
STRAT	Determines the strategy for computing the asymptotes	'S', 'N', 'Z', 'P', 'F'
ACTRES	Determines the active constraint set strategy; Constraints with values < ACTRES are active	Any real number

Table 3.1 User-specified Parameters for the SCP Algorithm

$$0.1 \leq t1 \leq 0.99$$
$$1 \leq t2 \leq 10 \ ,$$

which are also specified by the user.

Strategy S: A slightly modified scheme of Svanberg's original proposal

For $k = 0$ and $k = 1$,

$$L_i^k = \underline{x}_i - 0.1(\overline{x}_i - \underline{x}_i) \text{ and } U_i^k = \overline{x}_i + 0.1(\overline{x}_i - \underline{x}_i) \ .$$

For $k \geq 2$,

If $\text{sign}(x_i^{(k)} - x_i^{(k-1)}) \neq \text{sign}\ (x_i^{(k-1)} - x_i^{(k-2)})$,

$$L_i^k = x_i^{(k)} - 0.7(x_i^{(k-1)} - L_i^{k-1})$$
$$U_i^k = x_i^{(k)} + 0.7(U_i^{k-1} - x_i^{(k-1)}) \ .$$

If $\text{sign}(x_i^{(k)} - x_i^{(k-1)}) = \text{sign}\ (x_i^{(k-1)} - x_i^{(k-2)})$,

$$L_i^k = x_i^{(k)} - \frac{(x_i^{(k-1)} - L_i^{k-1})}{0.7}$$
$$U_i^k = x_i^{(k)} + \frac{(U_i^{k-1} - x_i^{(k-1)})}{0.7} \ .$$

Strategy N: Currently used by Svanberg

For $k = 0$ and $k = 1$,

$$L_i^k = x_i^{(k)} - \gamma(\overline{x}_i - \underline{x}_i) \text{ and } U_i^k = x_i^{(k)} + \gamma(\overline{x}_i - \underline{x}_i) .$$

For $k \geq 2$,

If $\text{sign}(x_i^{(k)} - x_i^{(k-1)}) \neq \text{sign }(x_i^{(k-1)} - x_i^{(k-2)})$,

$$L_i^k = x_i^{(k)} - t1(x_i^{(k-1)} - L_i^{k-1})$$
$$U_i^k = x_i^{(k)} + t1(U_i^{k-1} - x_i^{(k-1)}) .$$

If $\text{sign}(x_i^{(k)} - x_i^{(k-1)}) = \text{sign }(x_i^{(k-1)} - x_i^{(k-2)})$,

$$L_i^k = x_i^{(k)} - \frac{(x_i^{(k-1)} - L_i^{k-1})}{t2}$$
$$U_i^k = x_i^{(k)} + \frac{(U_i^{k-1} - x_i^{(k-1)})}{t2} .$$

Strategy Z: Used when variable bounds are not "reasonable"

Note that the values $lb_i^o(ub_i^o)$ denote the ith lower (upper) bound of the subproblem for x_i^o.

For $k = 0$,

If $|x_i^o| \leq 1$,

$$L_i^o = \max(-2, \underline{x}_i - 1) \text{ and } U_i^o = \min(2, \overline{x}_i + 1) .$$

If $|x_i^o| > 1$,

$$L_i^o = \max(-2|x_i|, \underline{x}_i - |x_i| - 1) \text{ and } U_i^o = \min(2|x_i|, \overline{x}_i + |x_i| + 1) .$$

For $k = 1$,

If $x_i^1 = ub_i^o$,

$$L_i^1 = L_i^o \text{ and } U_i^1 = \min(\max(10|x_i^1|, 2), \overline{x}_i + |x_i| + 1) .$$

If $x_i^1 = lb_i^o$,

$$L_i^1 = \max(\min(-10|x_i^1|, -2), \underline{x}_i - |x_i| - 1) \text{ and } U_i^1 = U_i^o .$$

Otherwise

$$L_i^1 = L_i^o \text{ and } U_i^1 = U_i^o .$$

For $k \geq 2$, same as in strategy S.

Strategy P: Must have lower bounds and initial guess greater than 0

$$\text{For all k : } L_i^k = t1\ x_i^k \text{ and } U_i^k = x_i^k / t1 .$$

Strategy F: An approximation of CONLIN

$$\text{For all k : } L_i^k = 0 \text{ and } U_i^k = 10^5 .$$

In this study, selection of appropriate strategies are determined through initial testing on a small test model. This is discussed more in-depth in Section 3.6.

3.4.3 Auxiliary Subproblem. A common difficulty with sequential convex programming methods occurs when an approximate subproblem must be built about a design point that is infeasible, which occurs frequently during the initial iterations. If two or more of the violated constraints are incompatible, an infeasible, and therefore unsolvable, subproblem is generated [12]. The SCP algorithm handles this situation by forming an auxiliary problem that relaxes the violated constraints by adding artificial variables [53]. For large-scale applications, this may significantly increase the complexity of the problem because the number of added artificial variables equals the number of violated constraints. This study monitors the impact of infeasible designs on algorithm performance and attempts to mitigate any negative impact to efficiency through the use of an effective constraint retention strategy, details of which are discussed in Section 3.6.

3.5 Integration of SCP into ASTROS

Since the SCP algorithm has the capability to construct the subproblem, it was not necessary to build this capability into DESIGN2. The primary requirements of the DESIGN2 module were,

- Bring the current design information (design variable values, active constraint values, active constraint gradients, objective function value, objective function gradient) into core and establish arrays to store the information.
- Initialize the parameters needed by the SCP algorithm.
- Call SCP to perform the redesign task.
- Update the ASTROS database with the new design information after subproblem convergence.

The FORTRAN subroutine that implemented DESIGN2 is listed in Appendix B.

One complication with this implementation is that SCP was developed with the capability to control the master loop of the optimization. As part of this capability, SCP can flag the active constraints according to a limit set on constraint values by the ACTRES parameter and call for the gradient evaluations of only those constraint functions that are considered active. These capabilities overlap the capabilities that are built into the standard ASTROS design. In this study it was desired to minimize the changes to the ASTROS standard loop so the research could focus more on how well SCP was solving the subproblems. As such, it was necessary to "trick" SCP into solving one subproblem at each master loop iteration by setting the maximum iteration parameter (for a master loop) within SCP to unity and setting the ACTRES parameter to a very high number. This implementation allowed ASTROS to flag active constraints and compute the necessary gradients, and SCP to construct and solve the subproblems. The appropriate arrays containing the design information as well as a logical array containing the active constraint flags were then passed to SCP when it was called by DESIGN2.

An additional complication with this implementation is that the line search option in the SCP algorithm requires evaluation of the *original* constraint functions and the objective function along the search direction, where the search direction is determined by the solution to the approximate convex subproblem. Obtaining these constraint values requires another FEA during the redesign phase, a departure from the standard ASTROS optimization loop. Implementation of this capability requires a significant coding effort. Additionally, it is doubtful that the benefits gained in the convergence properties of the algorithm would outweigh the added computational expense, particularly for the large problems considered by this study. Method 'A' (MMA with line search) was therefore not tested.

3.6 Design of the Investigation

SCP was implemented as the optimizer to perform the redesign task in ASTROS. This implementation was tested on three large-scale structural design models provided by the Air Vehicles Directorate/Structures Division of the Air Force Research Laboratories. The three chosen models were,

1. 200 member plane truss (Tr200),
2. Intermediate complexity wing (ICW), and
3. High-altitude, long-endurance (HALE) aircraft.

The Tr200 model, which has 200 design variables and 2500 constraints, was chosen because it is a classic large problem with known solutions. The ICW model has 350 design variables and 750 constraints. It was chosen because it has constraints from both the structural and flutter disciplines. The HALE model, having 1527 design variables and 6124 constraints, has been solved only twice, once by an optimality criteria method [9] and once using sequential quadratic programming [2]. It was chosen because it was the largest problem available. A more detailed description of these models is given in Appendix A.

Before testing the large models, appropriate choices for the following strategies were necessary:

- Asymptote determination
- Constraint retention.

The asymptote determination strategy is important because this determines the form taken by the subproblems. A poor choice for this strategy can lead to poor algorithm performance or even a divergent iteration sequence. The constraint retention strategy is important because infeasible designs are anticipated early in the optimization process. For the large problems being tested, too many violated constraints included in the subproblem may overwhelm the auxiliary subproblem and lead to a break down in the optimization. Conversely, enough constraints need to be retained to produce subproblems of high quality.

As presented in Section 3.4, the asymptotes L_i and U_i are determined in the SCP algorithm by selecting one of five options for the SCP parameter STRAT (S, N, Z, P, or F) and specifying a value for the applicable supporting parameters $t1$, $t2$, and γ. The number of retained constraints can be controlled either by specifying values for the ASTROS parameters **NRFAC** and ε or through the SCP parameter ACTRES. For this study, the constraint retention strategy was determined using the ASTROS parameters to allow for greater control over how many constraints were retained. The ACTRES parameter was set to a high number so that SCP would not decrease the number of retained constraints already specified through the ASTROS parameters.

In short, choices were required for six different parameters: STRAT, $t1$, $t2$, γ, **NRFAC**, and ε. Prudent use of computational resources required some preliminary analysis with a small test problem to obtain reasonable choices for the parameters. The ten-bar truss example of Figure 1.1, having only ten design variables and 18 constraints, was used for this purpose.

Figure 3.3 describes the sequence of testing conducted. A set of 12 experimental runs, at varying levels for the parameters, was executed using the ASTROS-SCP implementation. The results were analyzed to select appropriate parameter settings for the asymptote determination parameters. After these tests, initial tests on the Tr200 and ICW models

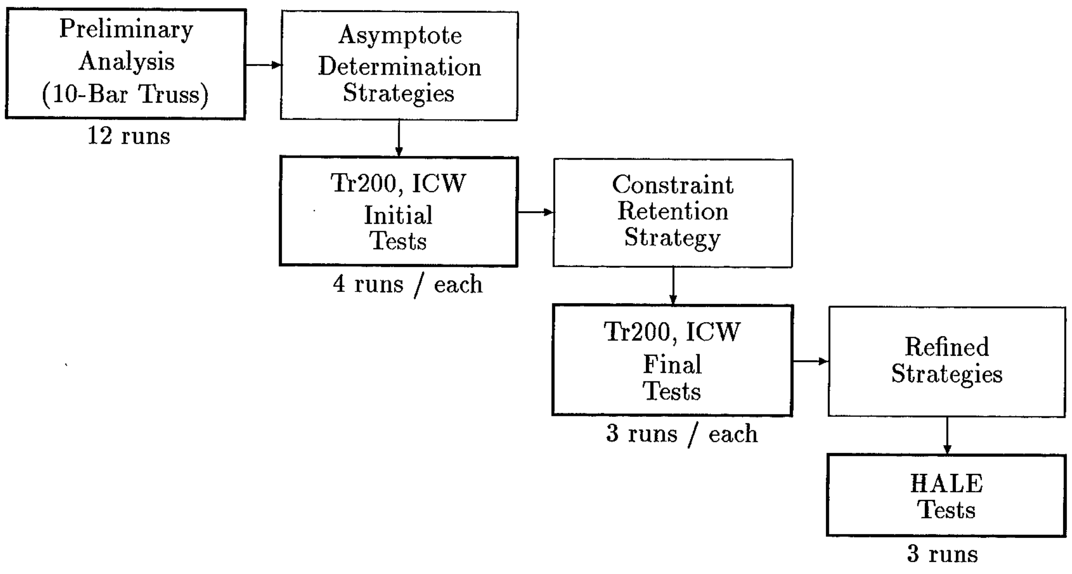


Figure 3.3 Design of the Investigation

were conducted to obtain appropriate settings for the constraint retention parameters, followed by the final test runs on these models. The last set of tests, conducted on the HALE model, used lessons learned from the Tr200 and ICW testing to refine both the asymptote determination and constraint retention strategies.

3.7 Summary

The details of the ASTROS-SCP implemention and the design of the investigation have been described in this chapter. The final test runs on the Tr200, ICW, and HALE models were attempted using both the traditional ASTROS loop and the alternative ASTROS-SCP implementation. Comparisons were made with respect to solution quality, CPU time, number of iterations required, and number of gradients computed. Results of the testing conducted is presented in the next chapter.

IV. Results

4.1 Overview

This chapter reports the results of the testing described in Section 3.6. The ASTROS-SCP implementation was first tested against the ten-bar truss finite-element model to obtain appropriate settings for the asymptote determination parameters. To arrive at an adequate constraint retention strategy, a series of four initial test runs were then executed on both the Tr200 and ICW models using a consistent asymptote determination strategy but a varying constraint retention strategy. These results were used to determine a constraint retention strategy for the remaining two runs. This initial testing is not conducted on the HALE model due to its large size. Lessons learned from the Tr200 and ICW initial test runs are used to estimate an appropriate constraint retention strategy for HALE. Three "final" runs were then executed on the HALE model.

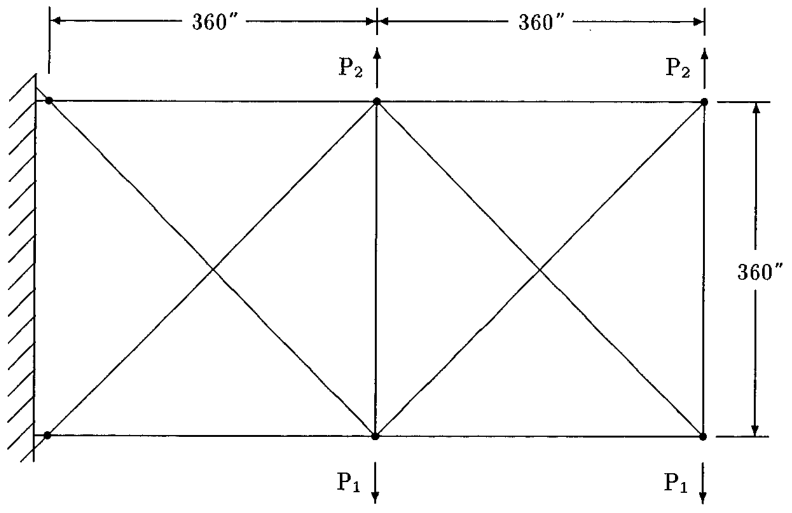
The three large test problems were also solved (or attempted) using the standard ASTROS approach. Three different constraint retention strategies are used in the ASTROS runs so that consistent results may be compared to the ASTROS-SCP implementation.

Each of the models is treated in individual sections. Results are summarized in tables and run charts. For the tables in this chapter, the following notation is used:

NRET	=	Number of constraints retained per iteration
NITER	=	Number of iterations required
F	=	Objective function value (weight in pounds)
CPU-TOTAL	=	Total CPU time required to terminate (seconds)
CPU-REDESIGN	=	Average CPU time spent in the redesign module (seconds)
NCG	=	Total number of individual constraint gradients computed
MCV	=	Maximum constraint violation.

4.2 Ten-Bar Truss

4.2.1 Model Description. The ten-bar truss model consists of eight nodes connected by ten finite elements representing structural rods. The design variables are defined



Material:	Aluminum, E=10^7 psi, ρ=.1 pci
Stress Limits:	±25000 psi (All members)
Displacement Limits:	±2.0 in (All nodes)
Lower Bounds:	6.67 $\times 10^{-3}$ in^2
Upper Bounds:	1000 in^2
Loading Conditions:	P_1=150,000 lbs, P_2=50,000 lbs

Figure 4.1 Ten-bar Truss Model and Design Conditions

as the cross-sectional areas of the rods. The material properties, stress limits, displacement limits, design variable bounds, and loading conditions are presented in Figure 4.1 (data taken from [36]). The resulting mathematical model consists of ten design variables, 18 behavioral constraints (stress and displacements), and 20 side constraints. The objective is to minimize structural weight. The initial solution vector is feasible.

4.2.2 Results and Analysis. Twelve experimental runs were executed using this model. The parameter settings for each of the runs are depicted in Table 4.1. The constraint retention strategy was controlled through the use of the ε parameter only; **NRFAC** was set to a value of zero for each of the runs. When ε is set to a very low number, *e.g.* -1000.0, all constraints are retained. When ε is set to near zero, *e.g.* -0.10, only the active and violated constraints are retained. The purpose of using these two settings is to compare the worst-case scenario (retention of all constraints) to an active set strategy.

A summary of the numerical results is presented in Table 4.2. Figure 4.2 shows the path of the objective function value for strategies Z, F, and P (at all three levels of $t1$) at

Run	Parameter settings				
Number	STRAT	t1	t2	γ	ε
1	S	-	-	-	-1000.0
2	N	0.8	1.2	0.5	-1000.0
3	Z	-	-	-	-0.1
4	Z	-	-	-	-1000.0
5	P	0.9	-	-	-0.1
6	P	0.9	-	-	-1000.0
7	P	0.5	-	-	-0.1
8	P	0.5	-	-	-1000.0
9	P	0.1	-	-	-0.1
10	P	0.1	-	-	-1000.0
11	F	-	-	-	-0.1
12	F	-	-	-	-1000.0

Table 4.1 Experimental Runs for Ten-bar Truss Model

a setting of -0.1 for ε. Figures 4.3 and 4.4 show the number of violated constraints and the maximum constraint violation histories, respectively, for strategies Z, F, P ($t1 = 0.1$), and P ($t1 = 0.5$) at the same setting for ε. Strategy P ($t1 = 0.9$) is omitted because each solution in the iteration sequence is feasible.

The data was analyzed to help select an asymptote determination strategy for use in testing the large problems. It was hoped that the data would also help determine a constraint retention strategy, but Table 4.2 shows that altering the constraint retention parameter (ε) has little to no effect on the solution time even though the number of individual gradients computed increases significantly when all the constraints are retained. For the large problems, it is anticipated that infeasible designs with many violated constraints will cause problems with algorithm performance and solution quality, necessitating the initial testing of the Tr200 and ICW models to arrive at an appropriate constraint retention strategy.

Table 4.2 indicates that strategies S and N produced a divergent series of solution vectors. This result is not surprising since, in each strategy, the asymptotes are computed using the difference of the upper and lower bounds on the design variables, which is on the order of 10^3 for this problem. As a result, poor choices are made for the asymptotes during subproblem construction, producing approximations of low quality.

Run	NITER	F	NCG	CPU-TOTAL
1		Divergent Series		
2		Divergent Series		
3	60[a]	5048.0[b]	304	118.0
4	60[a]	5048.0[b]	1062	117.5
5	60[a]	5704.5	110	114.2
6	60[a]	5702.6	1062	117.1
7	24	5097.2	44	51.9
8	23	5111.6	396	51.3
9	12	5085.0	26	31.4
10	12	5080.4	198	32.2
11	14	5087.5	91	35.2
12	10	5078.1	162	28.8

[a]max iterations
[b]infeasible design

Table 4.2 Ten-bar Truss Numerical Results

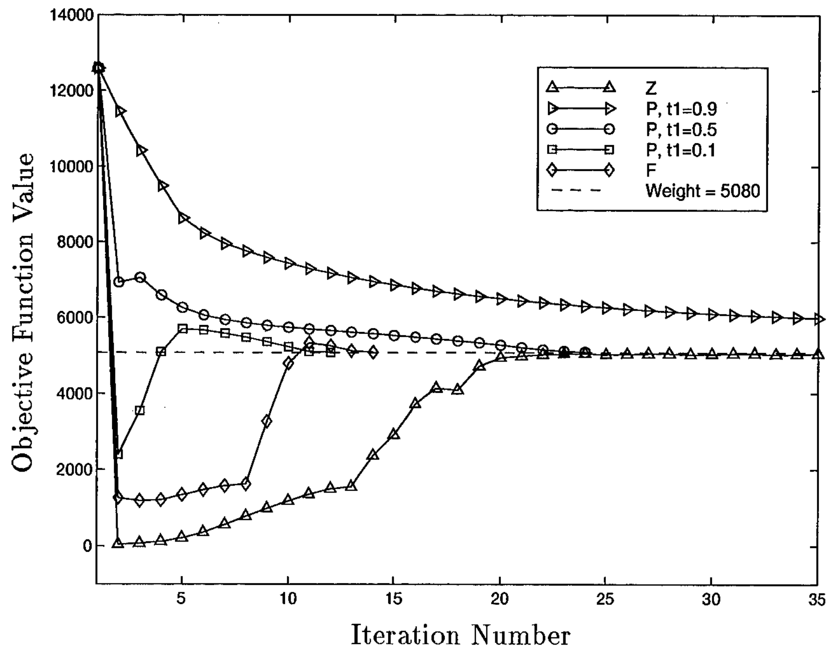


Figure 4.2 Ten-bar Truss Objective Function History (ε=-0.1)

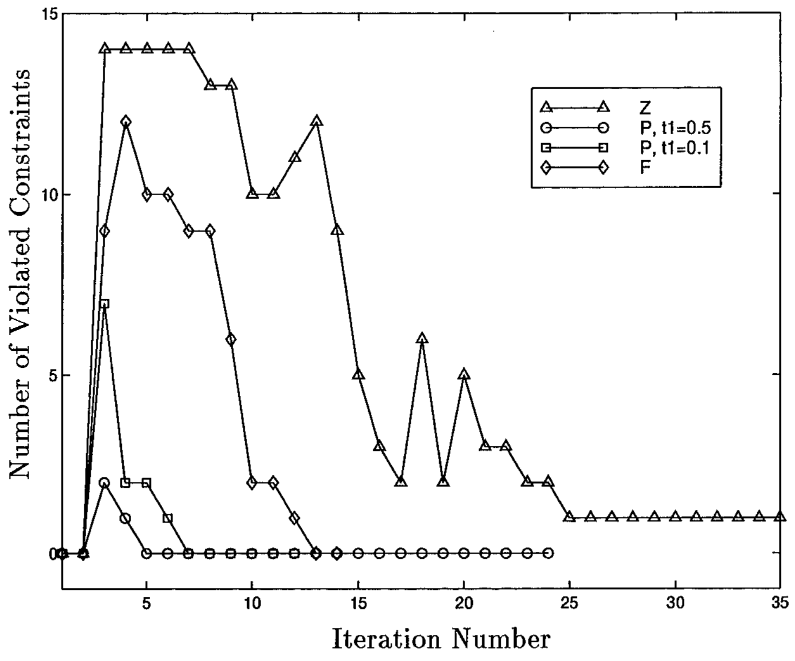


Figure 4.3 Ten-bar Truss Violated Constraint History (ε=-0.1)

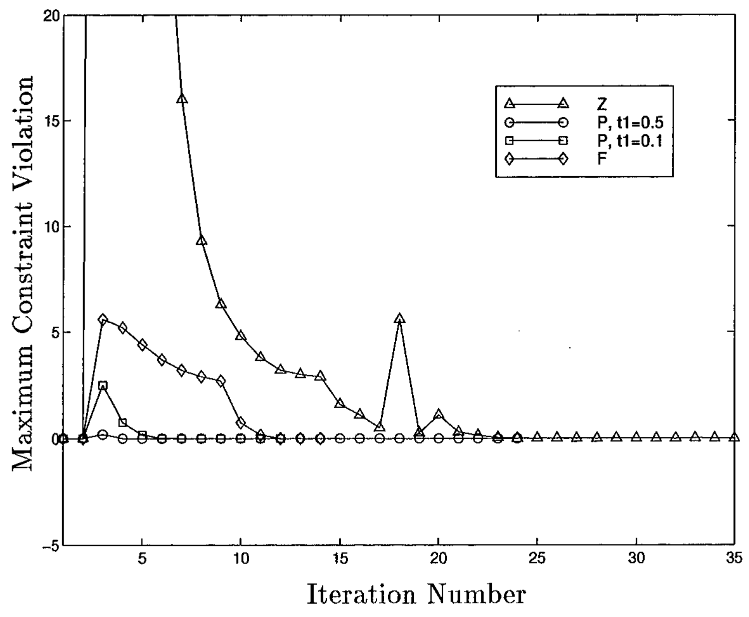


Figure 4.4 Ten-bar Truss Maximum Constraint Violation History (ε=-0.1)

Strategy Z quickly moves to a highly infeasible point and progresses through the infeasible region for 20 iterations until it reaches the neighborhood of the minimum-weight solution. The algorithm does not converge but rather oscillates between a series of four infeasible designs until reaching the maximum number of iterations. Other than at the initial design, this strategy never produces a feasible solution. Conversely, strategy P ($t1 = 0.9$) stays in the feasible region but progresses at such a slow rate towards the minimum weight that the maximum number of iterations is reached before nearing the minimum-weight solution.

As with strategy Z, strategies F, P ($t1 = 0.1$), and P ($t1 = 0.5$) also enter the infeasible region immediately but to a much lesser degree of infeasibility. Although P ($t1 = 0.5$) obtains feasibility sooner than the other two, it takes longer to converge to the minimum-weight solution. F and P ($t1 = 0.1$) are superior to the other strategies in terms of solution quality.

From the results it is clear that as the asymptotes are moved further away from the design variables, there is a greater potential for large changes in the solution. This is a consequence of the approximations that are built by each strategy. The further away the asymptotes are from the design variables, the less conservative the approximation, leading to a larger feasible region for the subproblem. For strategy P, moving the asymptotes further away from the design variables corresponds to a decrease in $t1$. Strategy F is merely a special case of strategy P in which the asymptotes are moved the furthest. Strategy Z enters the infeasible region immediately because at the first iteration the asymptotes are moved further away from the design variables than for the remaining strategies. The solution to the resulting subproblem represents the largest change in the redesign. Although large changes in the redesign can lead to infeasible designs during the early iterations, they can also lead to faster convergence, as was the case for Strategies P ($t1 = 0.1$) and F. In choosing the parameter settings a tradeoff is necessary between the potential for infeasible designs and the potential for convergence in fewer iterations.

For the large test problems, the initial solution vector and the design variable bounds are on the same order of magnitude as the ten-bar truss, and therefore strategies S and N can be eliminated from consideration. Strategy Z is eliminated because of its path through the infeasible region and its poor convergence. For strategy P, the choices for $t1$ are limited to values of less than 0.9 so that convergence is accelerated. For testing the Tr200 and ICW, the following strategies were used:

- P with $t1 = 0.5$,
- P with $t1 = 0.1$, and
- F.

Before testing HALE, the results from the Tr200 and ICW runs were analyzed to see if any adjustments should be made to the strategy selection.

4.3 200 Member Plane Truss (Tr200)

4.3.1 Model Description. This model consists of 77 nodes connected by 200 finite elements representing steel rods. The design variables are the cross-sectional areas of the rods. Stress constraints are placed on the rods and displacement constraints on the nodes. The model is subjected to five loading conditions. The resulting mathematical model consists of 200 design variables and 2500 behavioral constraints. The objective is to minimize structural weight. The initial solution vector is infeasible. Additional supporting data and a model diagram are included in Appendix A.

4.3.2 Initial Testing. Four test runs were conducted using Strategy P ($t1 = 0.1$) with varying constraint retention strategies. In three of the runs the number of constraints retained was determined explicitly by setting ε to a high number (1000.0) and specifying a value for **NRFAC**. The fourth run tested an active constraint set strategy in which all constraints with values greater than -0.10 were retained. Table 4.3 summarizes the results of these four runs and Figure 4.5 displays the cumulative CPU time per iteration for the first three runs.

Since the initial solution vector is infeasible, the SCP algorithm creates an auxiliary subproblem during the first few iterations. Using the active set strategy, 666 active and violated constraints are retained and the algorithm is overwhelmed by the size of the auxiliary subproblem, spending twenty hours of CPU time attempting to solve it. The

	Constraint Retention Strategy			
	NRFAC=0.25	NRFAC=0.50	NRFAC=0.75	NRFAC=0.0
ASTROS-SCP	ε=1000.0	ε=1000.0	ε=1000.0	ε=-0.10
NRET	50	100	150	666[a]
NITER	17	18	5[b]	3[b]
F	29413.6	29353.6	8460.9	9963.4
CPU-TOTAL	182.2	513.6	1392.1	39.99 hrs
CPU-REDESIGN	4.6	17.4	338.5	19.99 hrs
NCG	803	1702	604	1998

[a]number of active and violated constraints at initial design
[b]terminated early

Table 4.3 Tr200 Initial Test Results

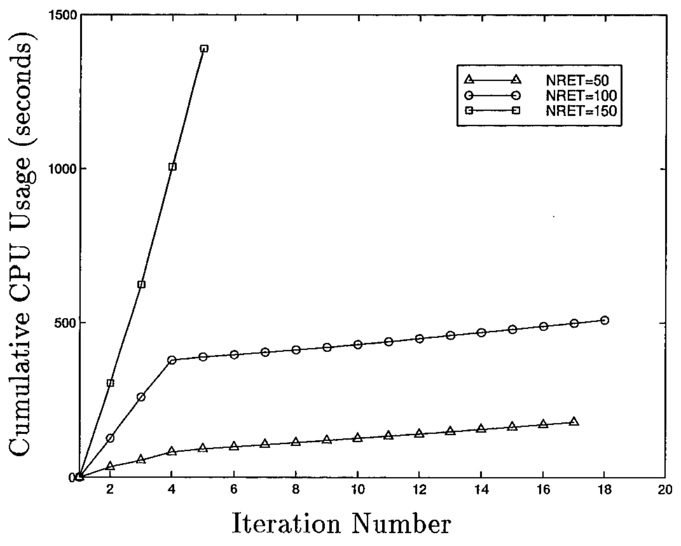


Figure 4.5 Tr200 Initial Test: CPU vs. Iteration Number

subroutine eventually returns an error message indicating the algorithm cannot overcome the infeasibility of the subproblem. Because ASTROS terminates with an error if the solution does not change after three iterations, the run completes after the third iteration.

Retaining 150 constraints per iteration, SCP solves the auxiliary problem in the first two iterations, returning infeasible designs each time. In the next iteration, however, the algorithm cannot overcome the infeasibility and ASTROS again terminates with errors after three more iterations.

In the remaining test runs, SCP overcomes this situation and the optimization eventually converges to the minimum-weight solution. In each case, a feasible design is reached at the fourth iteration. Figure 4.5 shows that the negative impact to computational efficiency is more pronounced for NRET = 100 than for NRET = 50 during the first three iterations. These results suggest the necessity for an aggressive constraint retention strategy to overcome the situation that an infeasible subproblem is generated. In testing strategies P ($t1$ = 0.5) and F, 50 constraints per iteration were retained.

4.3.3 Optimization Results. The results of testing the remaining strategies are summarized in Table 4.4. The objective function and maximum constraint violation histories are shown in Figures 4.6 and 4.7, respectively. The model was also solved three times using the standard ASTROS approach using the following constraint retention strategies:

- ASTROS default – **NRFAC** = 3.0 and ε = -0.10,
- active constraint set – **NRFAC** = 0.0 and ε = -0.10, and
- same strategy as ASTROS-SCP – **NRFAC** = 0.25 and ε = 1000.0.

Results are tabulated in Table 4.5. A convergence comparison plot is shown in Figure 4.8.

Although strategy F reaches the minimum-weight solution in fewer iterations, its computational performance is inferior to strategy P ($t1$ = 0.1). This is because strategy F requires eight iterations to reach feasibility where P ($t1$ = 0.1) needs only four. In fact, strategy F spends an average of 420 seconds performing the redesign during iterations 7 and 8 (compared to about 9.5 seconds for the first six iterations).

ASTROS-SCP	Strat P t1=0.5	Strat P t1=0.1	Strat F
NITER	34	17	15
F	29511.1	29413.6	29412.4
MCV	0.0	0.0	0.0
CPU-TOTAL	369.3	182.2	773.0
CPU-REDESIGN	4.8	4.6	48.3
NCG	1656	803	702

Table 4.4 ASTROS-SCP Tr200 Optimization Results (**NRFAC**=0.25, ε=1000.0)

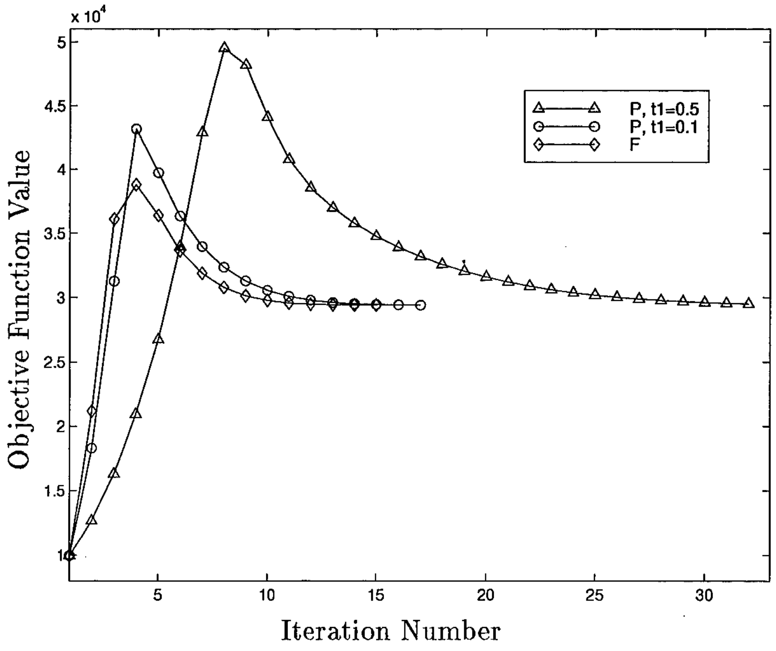


Figure 4.6 Tr200 Objective Function History (**NRFAC**=0.25 and ε=1000.0)

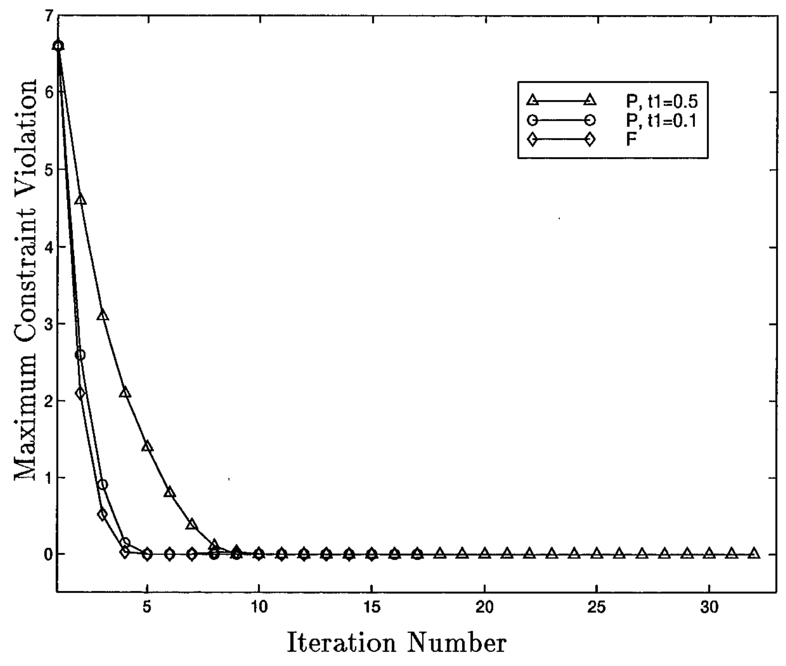


Figure 4.7 Tr200 Maximum Constraint Violation History (**NRFAC**=0.25 and ε=1000.0)

Strategy P ($t1$ = 0.5) does not encounter computational difficulties with infeasibility, but simply takes too many iterations to converge. Its solution quality is also inferior to the other two strategies.

The fastest of the standard ASTROS runs occurs using the same constraint retention strategy as ASTROS-SCP, converging to a slightly better solution in nearly half the time and computing fewer gradients than strategy P ($t1$ = 0.1). The other two ASTROS runs reach a lower weight solution but take more than twice as long as ASTROS-SCP, strategy

Standard ASTROS	Constraint Retention Strategy Default	 NRFAC=0.0 ε=-0.10	 NRFAC=0.25 ε=1000.0
NITER	15	18	14
F	29103.2	29153.6	29400.9
MCV	0.0	0.0	0.0
CPU-TOTAL	595.7	446.2	99.4
CPU-REDESIGN	34.4	20.2	1.1
NCG	8468	2069	651

Table 4.5 ASTROS Tr200 Optimization Results

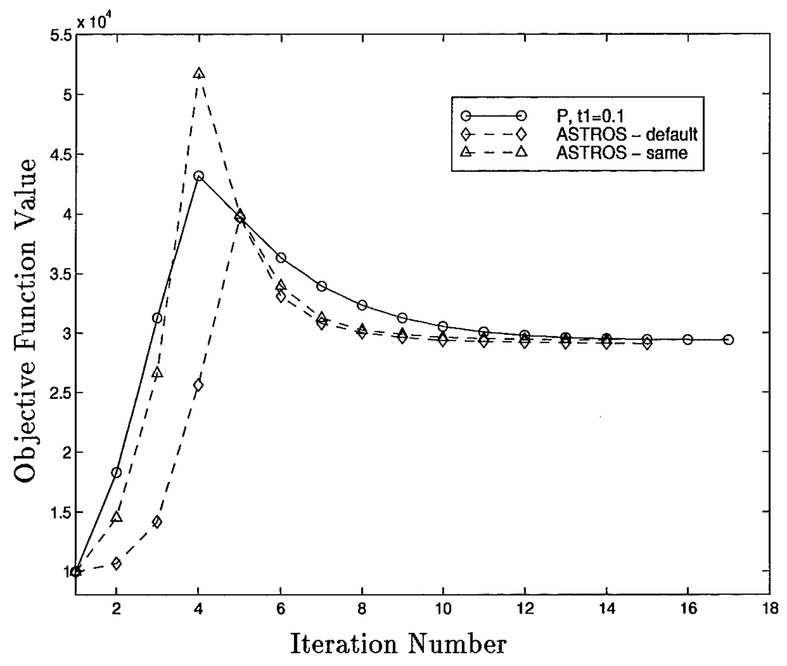


Figure 4.8 Tr200 Convergence Comparison

P ($t1 = 0.1$). Figure 4.8 shows that the ASTROS runs representing the best solution quality and fastest termination converge more directly to the region of the minimum-weight solution than strategy P ($t1 = 0.1$).

4.4 Intermediate Complexity Wing (ICW)

4.4.1 Model Description. This model consists of 88 nodes connected by 158 rods. Quadrilateral and triangular membrane elements model the wing skins, and shear panels model the spars and ribs. Stress constraints are imposed on the rods and displacement constraints imposed at the tip of the wing in the transverse direction. A flutter speed constraint of 925 knots, corresponding to a flight condition of Mach 0.8 at sea level, is also applied. The resulting mathematical model consists of 350 design variables and 750 behavioral constraints. The objective is to minimize structural weight and the initial solution vector is feasible. Additional supporting data and a model diagram are included in Appendix A.

	Constraint Retention Strategy			
	NRFAC=0.125	NRFAC=0.25	NRFAC=0.50	NRFAC=0.0
ASTROS-SCP	ε=1000.0	ε=1000.0	ε=1000.0	ε=-0.10
NRET	44	88	175	Active[a]
NITER	27	27	27	100[b]
F	41.835	41.817	41.818	44.293
CPU-TOTAL	501.6	653.4	865.1	2664.5
CPU-REDESIGN	5.2	9.3	16.9	13.6
NCG	1146	2288	4550	5430

[a]varies per iteration
[b]max iterations

Table 4.6 ICW Initial Test Results

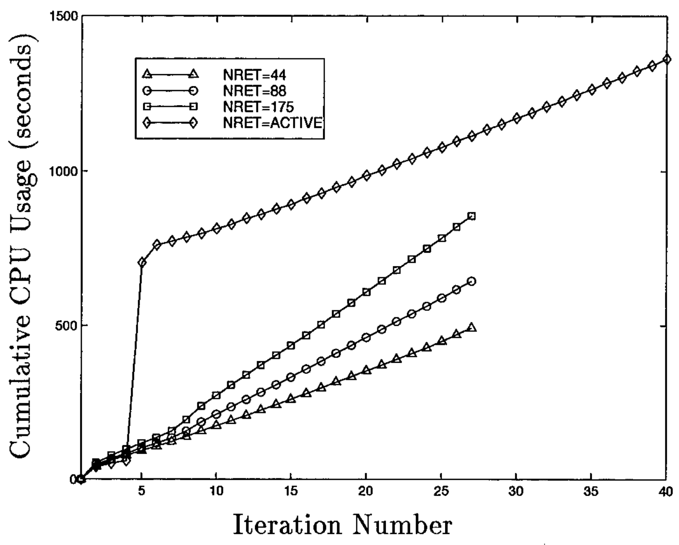


Figure 4.9 ICW Initial Test: CPU vs. Iteration Number

4.4.2 Initial Testing. As in the Tr200 tests, the ICW model was tested using strategy P ($t1 = 0.1$) at four different constraint retention strategies. Table 4.6 summarizes the results and Figure 4.9 shows the cumulative CPU time per iteration number. For this model, lower values of **NRFAC** were tested because the ratio of design variables to constraints is higher for ICW than Tr200.

In testing this model, the feasibility of the initial solution allowed each run to avoid infeasibilities in the early iterations. When using an active set strategy, however, the algorithm produced an infeasible point at the fourth iteration at which 185 constraints were active or violated. This not only impacted computational efficiency, as Figure 4.9

ASTROS-SCP	Strat P t1=0.5	Strat P t1=0.1	Strat F
NITER	30	27	100[a]
F	42.523	41.835	41.329
MCV	0.0	0.0	0.88514
CPU-TOTAL	534.6	501.6	40413.9
CPU-REDESIGN	4.2	5.2	392.9
NCG	1278	1146	4366

[a]reached max iterations

Table 4.7 ASTROS-SCP ICW Optimization Results (**NRFAC**=0.125, ε=1000.0)

shows, but also produced a series of infeasible designs that the algorithm could not recover from. The run eventually reached the maximum number of iterations before converging.

The efficiency of the dual solution method is demonstrated by Figure 4.9. Since the dual problem has the same dimension as the number of retained constraints, the constraint set strategy that retains only 44 constraints per iteration is most efficient. This strategy was used in testing the remaining asymptote determination strategies P ($t1 = 0.5$) and F even though the minimum-weight solution was slightly higher than when 88 constraints were retained.

4.4.3 Optimization Results. The results of testing the remaining strategies are summarized in Table 4.7. The objective function and maximum constraint violation histories are shown in Figures 4.10 and 4.11, respectively. As in the Tr200 tests, the model was also solved three times using the standard ASTROS approach under the ASTROS default constraint retention strategy, an active constraint set strategy, and a constraint retention strategy that mirrored the ASTROS-SCP approach. The results are tabulated in Table 4.8. A convergence comparison plot is shown in Figure 4.12.

Using strategy F, the iteration sequence enters the infeasible region after three iterations and cannot recover, eventually reaching the maximum number of iterations, even though the solution is in the neighborhood of the minimum weight (see Figure 4.10). The algorithm also uses close to seven minutes of CPU time per iteration, again demonstrating SCP's difficulties in handling infeasible design points for this strategy.

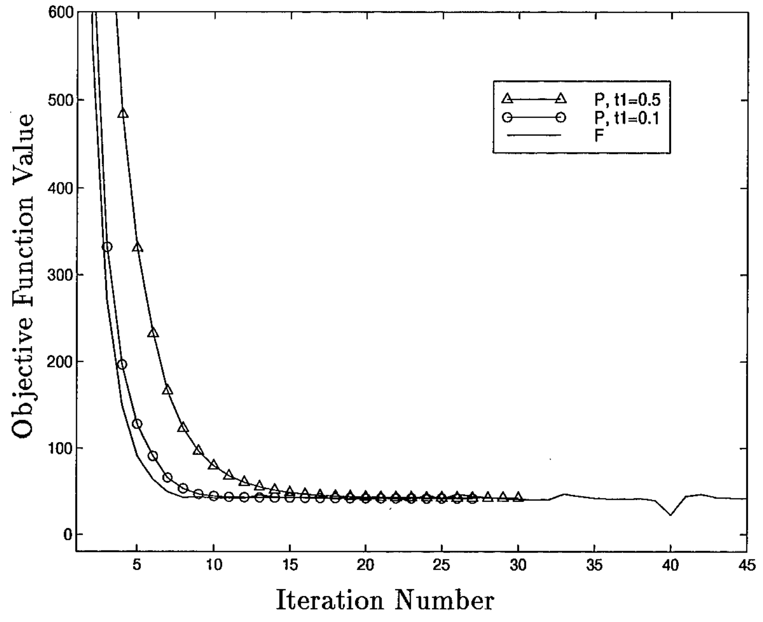


Figure 4.10 ICW Objective Function History (**NRFAC**=0.125 and ε=1000.0)

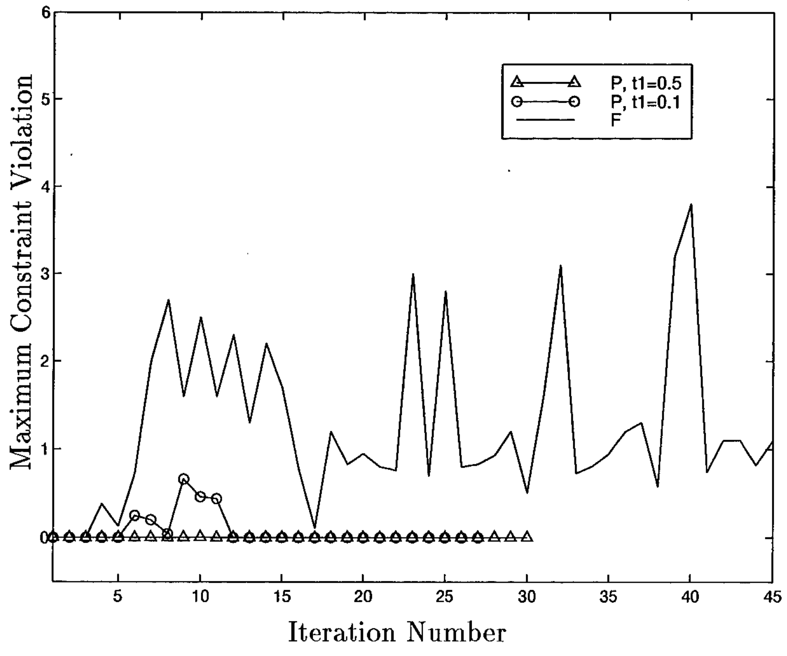


Figure 4.11 ICW Maximum Constraint Violation History (**NRFAC**=0.125 and ε=1000.0)

Standard ASTROS	Constraint Retention Strategy		
	Default	NRFAC=0.0 ε=-0.10	NRFAC=0.125 ε=1000.0
NITER	19	13	14
F	41.853	42.872	42.189
MCV	0.0	0.0	0.0
CPU-TOTAL	1135.8	373.4	326.9
CPU-REDESIGN	43.6	17.1	9.8
NCG	13500	421	572

Table 4.8 ASTROS ICW Optimization Results

By contrast, both versions of strategy P converge to a minimum-weight solution by producing a series of feasible, or nearly feasible, designs. These two strategies average about 18 seconds per iteration – less than six seconds performing the redesign. Strategy P ($t1$ = 0.1) produces the lowest structural weight design.

The most notable difference between the standard ASTROS results and the ASTROS-SCP results is that ASTROS solves in significantly fewer iterations. Figure 4.12 shows that the ASTROS-SCP reaches the neighborhood of the minimum-weight design in more iterations than ASTROS but again is slow to satisfy convergence criteria. However, ASTROS-SCP with strategy P ($t1$ = 0.1) achieves a lower weight and is more efficient performing the redesign.

4.5 High-Altitude, Long-Endurance (HALE) Aircraft

4.5.1 Model Description. The mission of this aircraft is to patrol for several days at 150-250 knots at an altitude of 65,000 feet. The model is made of a truss substructure and metallic cover skins. The design variables represent the cross-sectional areas of rods and the thicknesses of panels. Constraints are imposed on the member stresses and the wing-tip deflections. Four static loads are applied to simulate aerodynamic forces. The resulting mathematical model consists of 1527 design variables and 6124 constraints. The objective is to minimize structural weight. The initial solution vector is infeasible. Additional supporting data and a model diagram are included in Appendix A.

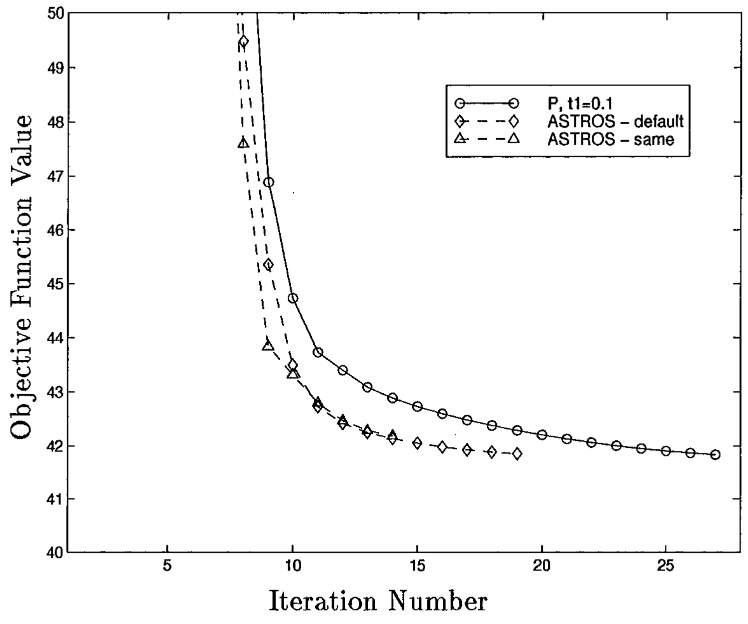


Figure 4.12 ICW Convergence Comparison

4.5.2 Optimization Results. On the Tr200 and ICW models, strategy F was the most computationally expensive of the three strategies tested and in the case of ICW, did not converge to a minimum-weight solution. In testing HALE, strategy F was replaced with strategy P ($t1 = 0.7$) in hopes of avoiding a non-converging sequence of infeasible design points. The tests on the Tr200 and ICW models also demonstrated that the number of retained constraints should be less than 100 to have a chance of recovering from infeasible design points. For the HALE test runs the value of **NRFAC** was set to 0.06 so that approximately 92 constraints would be retained at each iteration. The results of the test runs using strategy P at levels $t1 = 0.7$, $t1 = 0.5$, and $t1 = 0.1$ are summarized in Table 4.9. The objective function and maximum constraint violation histories are shown in Figures 4.13 and 4.14, respectively.

The HALE model could not be solved using the standard ASTROS optimization within the available computer memory. To solve HALE, ASTROS requires about 15 megawords of dynamic memory, primarily working memory for the μ-DOT subroutine. By comparison, the ASTROS-SCP implementation required less than one megaword. As a consequence, no ASTROS results are available for comparison. However, comparisons

ASTROS-SCP	Strat P t1=0.7	Strat P t1=0.5	Strat P t1=0.1
NITER	94	150[a]	35[b]
F	1453.06	1452.49	487.70
MCV	0.0	0.4	12.0
CPU-TOTAL (min)	836	1044	2990
CPU-REDESIGN (min)	8.2	6.3	84.4
NCG	8556	13708	3220

[a]reached max iterations before converging
[b]divergent series

Table 4.9 ASTROS-SCP HALE Optimization Results (**NRFAC**=0.06, ε=1000.0)

are made to solutions that have been reported in the literature on two occasions using non-standard approaches within ASTROS.

As the figures show, strategy P ($t1$ = 0.1) produces a large change in the solution vector immediately in the iteration sequence. The algorithm averages 84 minutes of CPU time in the redesign step but cannot recover from the severe infeasibilities. The optimization is finally halted after 35 iterations and nearly 50 hours of run time. Strategy P ($t1$ = 0.5) takes a much smoother path towards the minimum-weight solution but, interestingly, encounters minor infeasibilities (in terms of magnitude) at each iteration until reaching the maximum number of iterations. Strategy P ($t1$ = 0.7) progresses nicely from the initial infeasible design point to the feasible region after six iterations and converges to the minimum-weight design at iteration 94 in just under 14 hours of run time.

The minimum-weight solution of 1453.1 pounds obtained in this study is the lowest recorded to date. Using an optimality criteria approach, Canfield and Venkayya reported a feasible weight of 1650.6 pounds after 13 iterations [9]. Using a sequential quadratic programming implementation, Abramson and Chrissis reported a feasible weight of 1601.4 pounds after 91 iterations, but his algorithm terminated prematurely after exhausting computer memory [2]. The ASTROS-SCP implementation is the first MP method to successfully solve the HALE model to the convergence criteria.

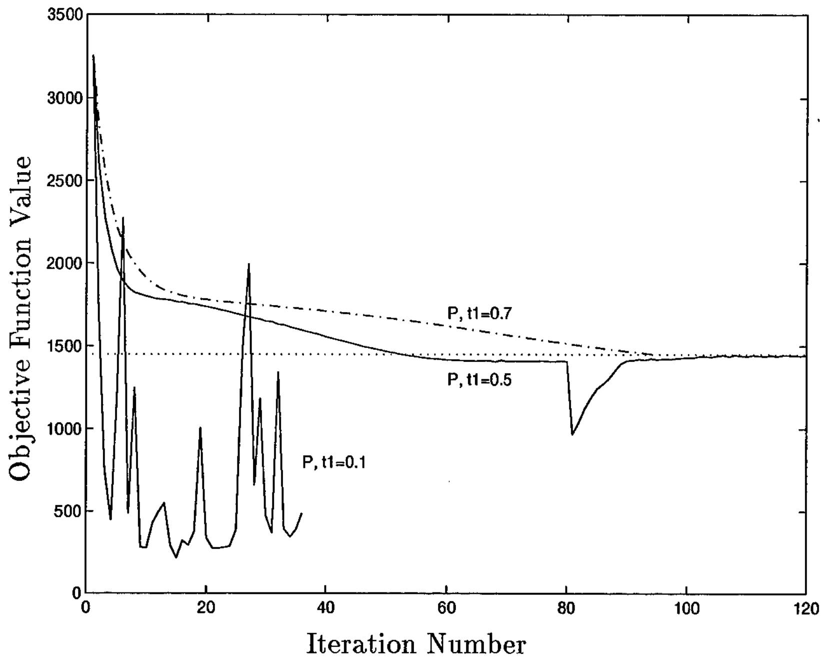


Figure 4.13 HALE Objective Function History

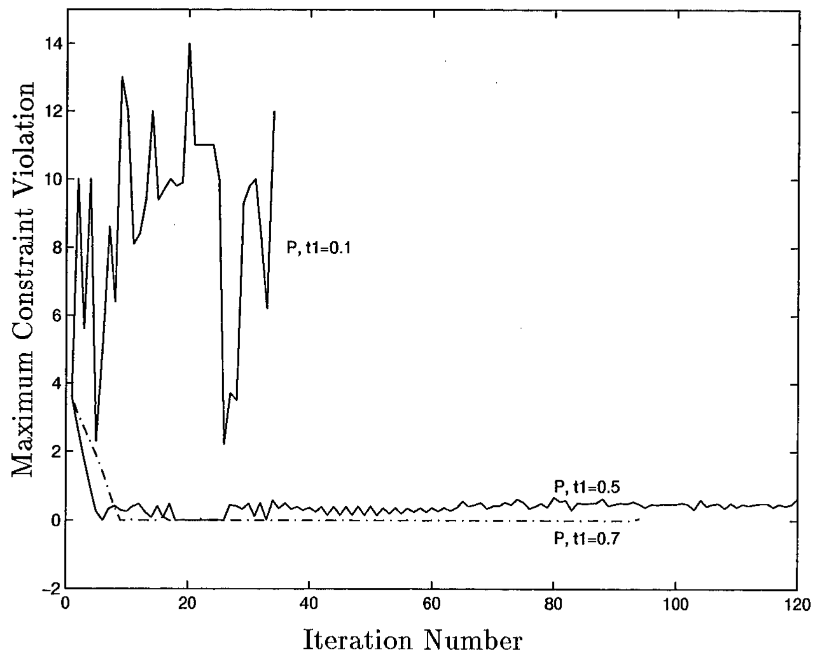


Figure 4.14 HALE Maximum Constraint Violation History

4.6 Summary

ASTROS-SCP successfully solved three large-scale structural design problems. Comparisons were made, where possible, to the solutions using the traditional ASTROS approach. The next chapter gives conclusions and recommendations for further research.

V. Conclusions and Recommendations

5.1 Summary

The purpose of this thesis was to apply a sequential convex programmming method to optimize large-scale structural design models in a multi-disciplinary environment. The literature suggests that these methods solve structural optimization problems efficiently but have not been rigorously tested on large problems with multi-disciplinary constraints.

Zillober's SCP algorithm was chosen as the optimizer primarily because of its flexible asymptote determination strategy and its FORTRAN implementation, which made it easily integrated into the ASTROS design tool. In a previous study, this algorithm had been shown to perform well on small to medium-sized structural optimization problems [34].

In the standard ASTROS approach, the method of modified feasibile directions solves a subproblem at each design iteration using approximate functions of the objective function and the retained constraints. For this research, SCP was integrated into the ASTROS optimization loop as the method to construct and solve the approximate subproblems. The hypothesis was that this approach, combined with an aggressive constraint retention strategy to take advantage of SCP's dual solution method, could lead to computationally efficient results.

After testing the ASTROS-SCP implementation against the low-dimensional ten-bar truss model to obtain appropriate parameter settings, the implementation was tested on three large-scale structural models, one with multi-disciplinary constraints. ASTROS-SCP successfully solved each model to the convergence criteria. For the Tr200 and ICW models, the results showed that the standard ASTROS approach was more efficient when the same constraint retention strategy was used. However, ASTROS-SCP reached a better solution for the ICW model.

The largest model tested was the HALE model. The standard ASTROS configuration could not solve HALE because the memory requirements exceeded available resources. Using the ASTROS-SCP implementation, the HALE problem was solved to the optimality conditions for the first time by a mathematical programming method within the ASTROS environment. The resulting design is a 9% improvement over the previously recorded minimum weight.

5.2 Conclusions

This research has demonstrated that SCP can be used to optimize large-scale structural designs. Analysis of the results leads to the following conclusions.

5.2.1 Convergence and Efficiency. The efficiency of the SCP dual solution technique is demonstrated by examining the redesign CPU averages of Tables 4.7 and 4.8 for the ICW model results. Strategy P, at both levels of $t1$, performs the redesign in about half the time of ASTROS (using the same constraint retention strategy). The same cannot be said for strategy F (see Table 4.7) nor any of the Tr200 tests (see Tables 4.4 and 4.5). Clearly, redesign efficiency is impacted in these runs by the infeasible design points that are encountered.

Tables 4.4, 4.5, 4.7, and 4.8 show that when using the ASTROS-SCP implementation combined with an aggressive constraint retention strategy, the Tr200 and ICW models were solved faster than with the standard ASTROS approach and the default constraint retention strategy. Using the same constraint retention strategy, ASTROS was able to solve the models faster but to a worse solution than when using less restrictive constraint retention strategies. In the case of ICW, ASTROS-SCP would have solved faster but simply could not converge to the optimal design point fast enough; the ASTROS solution converged in nearly half the number of iterations albeit to a slightly worse solution. Although ASTROS was faster, the ASTROS-SCP solution times of three minutes for Tr200 and eight minutes for ICW are well within what may be considered as reasonable.

For the HALE model, although no ASTROS results are available for comparison, the 14 hour run time that ASTROS-SCP required to solve to an indicated optimal solution (see Table 4.9) is well within reason for this size problem. As a comparison, on an older computing system, Abramson and Chrissis reported a 198 hour run time for three fewer iterations in solving HALE [2]. The attractive quality of ASTROS-SCP is the reduced memory requirement. The efficient use of dynamic memory by SCP suggests that problems of a much larger size could be solved using this implementation.

5.2.2 Multi-disciplinary Constraints. The inclusion of flutter constraints in the ICW model may have played a part in the slow convergence of ASTROS-SCP. The test run results indicate that, of the 44 constraints retained per iteration using strategy P ($t1 = 0.1$), 30 were flutter constraints in the first two iterations, 24 were flutter in the third iteration, and at least five were flutter from the fourth through the eighth iterations. This high proportion of retained flutter constraints may have led to poor approximations, causing a less-than-direct search to the region of optimality. More research is necessary in this area before conclusive statements can be made.

5.2.3 Constraint Retention. The results of the initial trial runs of Chapter IV show that the ASTROS-SCP performance was seriously impacted by the number of retained constraints at infeasible design points. In particular, the Tr200 results of Table 4.3 showed that too many retained constraints led to a break down in the optimization but the situation could be overcome by aggressively restricting the number of constraints. The best results employed a retention scheme whereby the number of retained constraints was a small fraction of the design variables. This strategy also takes advantage of SCP's efficient dual solution technique by limiting the number of variables in the dual problem to the number of active constraints.

5.3 Recommendations for Future Research

This study has shown that sequential convex programming methods are valid optimization methods for large-scale structural design models. The extent of their promise in this realm, however, is not completely understood. The remainder of this chapter proposes potential topics for further study that may help expand knowledge in this area.

Multi-disciplinary Model Testing. In testing the ICW model, this study considered a structural design with constraints from both structures and flutter. In the previous section, conclusions were postulated on the performance of the ASTROS-SCP implementation on this model, but this only represents an isolated case study. A research effort dedicated to the application of SCP methods to multi-disciplinary design models would shed more light on how sequential convex programming methods perform against a wider range of test problems. Such a study should choose test problems with a variety of representative disciplinary constraints so that lessons can be learned about which constraint types the convex approximation schemes have difficulties with.

Asymptote Determination. This study fixed the asymptote determination strategy for each test, but there is no reason the strategy cannot be altered at some point during the optimization. For example, an adaptive strategy that employs a more conservative strategy [such as strategy P ($t1 = 0.5$)] to force an initially infeasible design into the feasible region, followed by a less conservative strategy (such as F) to accelerate convergence to optimality, may lead to more efficient results. Since the choices for asymptotes are based almost completely on numerical experience, more empirical data is needed upon which to base judgements. There is much opportunity for computational experimentation in this area.

Constraint Retention. Analogous to the asymptote determination schemes, nothing forbids changing the strategy that chooses the number of constraints to retain at any given iteration. It is known that SCP methods are more efficient when fewer constraints are retained, but enough constraints need to be retained to construct an approximate subproblem of sufficient quality. Further study with constraint retention strategies may yield adaptive schemes that provide more efficient and/or more accurate solutions.

Extensions to the ASTROS-SCP Implementation. To improve the robustness of the ASTROS-SCP approach, steps could be taken to enable the line search option of the SCP algorithm. This would involve a coding effort to allow the DESIGN2 module to return to the ASTROS executive sequence and perform a finite-element analysis, then returning the function values to the DESIGN2 module. It is doubtful that computational efficiency would be gained with this modification, but the reliability of convergence could be improved. This can be important for large-scale problems because feasible initial solutions are difficult to ensure.

Recent Advances in Convex Approximations. Over the last three years, research has been directed toward the improvement and further generalization of convex approximation methods. In particular, Ma and Kikuchi propose an approximation that expands a Taylor-series about a new intermediate variable,

$$y_i = | x_i - c_i |^{\xi_i}$$

where ξ_i and c_i are parameters for the design variable x_i [25]. The authors label the approach as a further generalization to CONLIN and MMA but with enhanced flexibility and convergence properties.

Additionally, Zhang and Fleury propose a modified CONLIN method in response to the need for a general and flexible method that can be reliably applied to constraints of multiple disciplines [50]. The method adjusts the convexity of the CONLIN approximation using a two-point fitting scheme based on the available function value at the previous iteration. The scheme ensures that the current approximated function will exactly fit at the previous iteration design point. The purpose of the scheme is to expand the validity of the approximation over a larger region of the design space.

The ASTROS environment provides the ideal environment for testing of such methods. Its modularity provides enough flexibility to integrate alternative optimization methods and its powerful capabilities provide for analysis of large, multi-disciplinary designs.

Interior Point Methods for Convex Programming. The topic of interior point solution methods for convex programming problems has received considerable attention in recent years (see, for example [31] and [20]). In engineering analysis, these methods have been applied primarily to control theory but have also been used in the optimization of large-scale truss topology and shape designs [6]. The appeal of these methods is the efficient, polynomial-time algorithm performance. The application of interior point methods to the solution of a sequence of approximate convex subproblems is an area worthy of research in a structural optimization context.

Appendix A. Description of Test Problems

This chapter provides supporting data for the test models solved. A finite-element model diagram and table of design conditions are included for each.

A.1 200 Member Plane Truss

Table A.1 shows the design conditions on the Tr200 model [9]. Figure A.1 displays the finite-element diagram.

A.2 Intermediate Complexity Wing

Table A.2 shows the design conditions on the ICW model [9]. Figure A.2 displays the finite-element diagram.

A.3 High-Altitude, Long-Endurance Aircraft

Table A.3 shows the design conditions on the HALE model [9]. Figure A.3 displays the finite-element diagram.

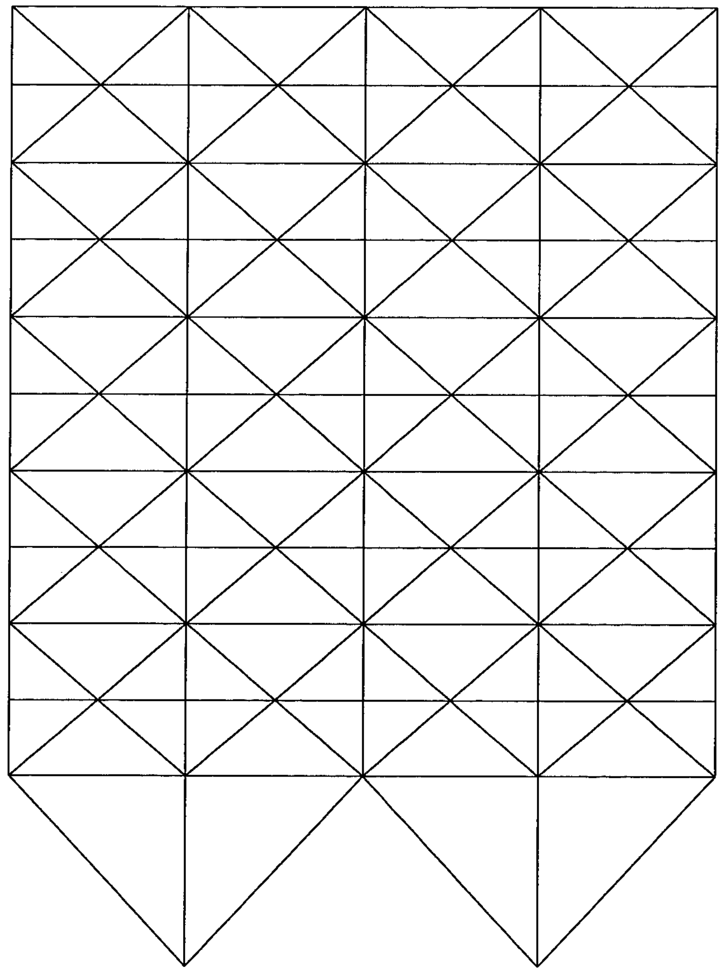


Figure A.1 200 Member Plane Truss Model

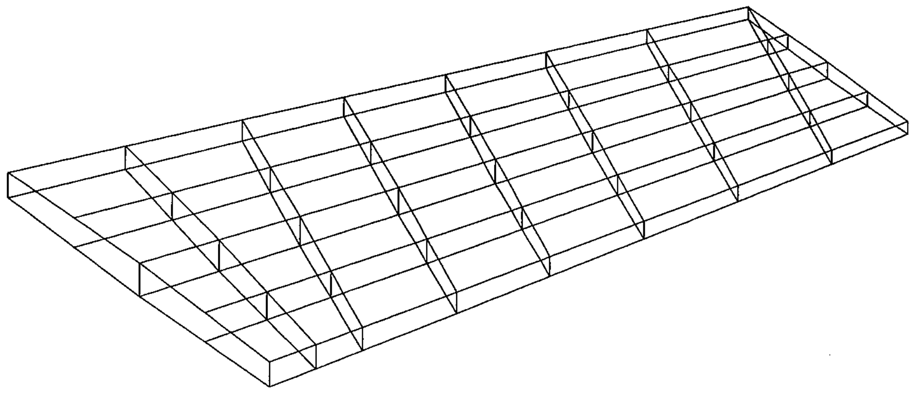


Figure A.2 Intermediate Complexity Wing Model

Table A.1 200-Member Plane Truss Design Conditions

Material, steel	
Modulus of elasticity	$E = 30 \times 10^6$ psi
Weight density	0.283 lb/in.^{3}
Stress limits	30,000 psi
Lower limit on rod areas	0.1 in.^{2}
Displacements on all nodes (horizontal and vertical directions)	0.5 in.
Number of loading conditions	5
Loading condition 1	1000 lb acting in $+X$ direction at nodes 1, 6, 15, 20, 29, 34, 43, 48, 57, 62, 71
Loading condition 2	1000 lb acting in -X direction at nodes 5, 14, 19, 20, 28, 33, 42, 47, 56, 61, 70, 75
Loading condition 3	10,000 lb acting in -Y direction at nodes 1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 15, 16, 17, 18, 19, 20, 22, 24, ..., 71, 72, 73, 74, 75
Loading condition 4	Loading conditions 1 and 2 together
Loading condition 5	Loading conditions 2 and 3 together

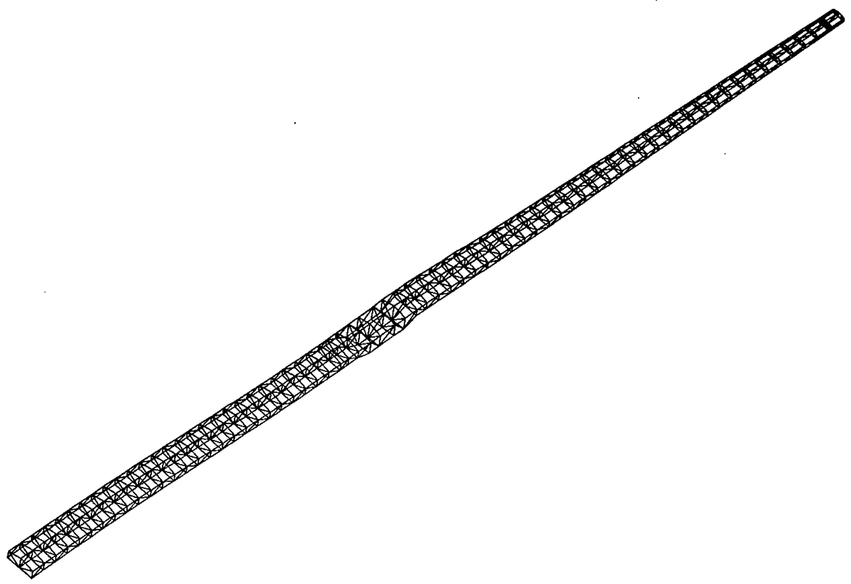


Figure A.3 High-Altitude, Long-Endurance Aircraft Model

Table A.2 Intermediate Complexity Wing Design Conditions

Isotropic material, aluminum	
Modulus of elasticity	$E = 30 \times 10^6$ psi
Poisson's ratio	0.30
Weight density	0.1 lb/in.^{3}
Tensile stress limit	67,000 psi
Comprehensive stress limit	57,000 psi
Shear stress limit	39,000 psi
Lower limit on thickness (shear panels)	0.02 in.
Lower limit on rod areas	0.02 in.^{2}
Orthotropic material, graphite epoxy	
Modulus of elasticity	$E_1 = 30 \times 10^6$ psi
	$E_2 = 1.6 \times 10^6$ psi
	$G_{12} = 0.65 \times 10^6$ psi
Poisson's ratio	0.25
Weight density	0.055 lb/in.^{3}
Stress limits	115,000 psi
Lower limit on plies	0.00525 in.
Behavior constraints	
Limit on transverse tip displacements	10.0 in.
Flutter speed limit	925 knots

Table A.3 High-Altitude, Long-Endurance Aircraft Wing Design Conditions

Isotropic material, aluminum	
Modulus of elasticity	$E = 10.5 \times 10^6$ psi
Poisson's ratio	0.30
Weight density	0.1 lb/in.^{3}
Stress limits	60,000 psi
Lower limit on thickness (shear panels)	0.021 in.
Lower limit on rod areas	0.10 in.^{2}
Behavior constraints	
Limit on transverse tip displacements	200.0 in.
Number of loading conditions	4

Appendix B. SCP Software Implementation

Implementing the alternate redesign task in ASTROS required the definition of a new ASTROS module. This module served as a driver program that called Zillober's SCP subroutine. The required ASTROS configuration changes are described in section B.1. The source code for the alternate redesign module, DESIGN2, is listed in section B.2. Zillober's SCP source code is omitted. The code was compiled in ASTROS version 13 on an IBM RS6000, Unix-based system at the Air Force Research Laboratory, Wright-Patterson Air Force Base.

B.1 ASTROS Configuration Changes

The following steps were taken to configure ASTROS such that the DESIGN2 module could be called to perform the redesign step in the ASTROS optimization. Note that the file containing the DESIGN2 subroutine was named design2.f and the SCP subroutine was named ascp.f.

Step 1: The DESIGN2 module and the SCP subroutine were compiled and linked by invoking the ASTROS FORTRAN 77 compiler with the following command:

```
% astrosf77 design2.f ascp.f
```

Step 2: So that ASTROS could recognize DESIGN2 as a module, the module was defined by adding the following lines to the end of the `MODDEF.DAT` file:

```
 DESIGN2   11
 102   1    1   4   2   2   2   7   7   7   8   7
C
C     PROCESS 'DESIGN2' - ALTERNATE REDESIGN MODULE
C
      CALL DESIGN2 ( IP(1), IP(2), LP(3), RP(4), RP(5), RP(6),
     1               EP(7), EP(8), EP(9), EP(10), EP(11) )
END
```

Step 3: All of the following files were included in the working directory:

MAPOLSEQ.DAT, MODDEF.DAT, RELATION.DAT,

SERRMSG.DAT, and TEMPLATE.DAT.

Step 4: ASTROS system parameters were defined in the local directory with the following command:

```
% sysgen
```

Step 5: An ASTROS version 13 was created in the local directory with the following command:

```
% makelocalastros
```

Step 6: The DESIGN2 module was called instead of the standard ASTROS design module through an edit to the standard MAPOL sequence. The following lines in the data input file accomplished this:

```
EDIT GO NOLIST
REPLACE 1887, 1889
CALL DESIGN2( NITER, NDV, APPCNVRG, CNVRGLIM,
             CTL, CTLMIN, GLBDES, CONST, CONSTORD,
             [AMAT], DESHIST);
```

This completed the steps necessary to override the ASTROS design module with DESIGN2. The tests were run by invoking the standard ASTROS execution command, `astros13_uai filename.d`.

B.2 DESIGN2 Module Source Code

```
      SUBROUTINE DESIGN2 (NITER, NDIM, APPCNVRG,
     +                    CNVRGLIM, CTL, CTLMIN, GLBDES,
     +                    CONST, CSTORD, A, DESHIST)
C****************************************************************
C   DESIGN2 MODULE:
C   THIS SUBROUTINE IS A DRIVER FOR AN ALTERNATE DESIGN TASK
C   TO THE STANDARD DESIGN MODULE IN ASTROS. THE SUBROUTINE
C   SOLVES THE APPROXIMATE PROBLEM AT EACH ASTROS DESIGN
C   ITERATION USING SEQUENTIAL CONVEX PROGRAMMING IMPLEMENTED
C   IN ZILLOBER'S SCP ALGORITHM. THE SCP SUBROUTINE IS CALLED
C   FROM THIS DRIVER MODULE.
```

```
C
C   AUTHOR:  CAPT TODD A. SRIVER, AFIT/ENS, GOR-98M, 08-FEB-98
C************************************************************
C
      CHARACTER*8 DSLIST(3)
      CHARACTER*8 DHLIST(11)
      CHARACTER*(*) A, GLBDES, CONST, CSTORD, DESHIST
      INTEGER     INFO(20),IXO,IX,IDX,IXU,IXL,IDF,IG,IDG,IU,
     +            IUL,IUU,IACTIVE,IDIMN,LDIMN,RDIMN,I,J,
     +            ICON,ITEMP,IACT,WA,MNN,IDV,ISTOBJ,POSITN(2)
C
      INTEGER NDIM,NMAX,MG,MH,MMAX,ITMAX,IPRINT,NOUT,
     +        RDIM,IDIM,LDIM,IERR,MODE,NRF,
     +        NRG,MAXINT,ITDUAL
      LOGICAL APPCNVRG
      DOUBLE PRECISION F,FO,ACC,ACTRES,T1,T2,GAMMA,EMACH
      CHARACTER*1 METHOD, STRAT
C
C   DYNAMIC MEMORY ALLOCATION: ALL DATA ARRAYS STORED IN CORE
C
      DOUBLE PRECISION DCORE(1)
      REAL RCORE(1)
      INTEGER ICORE(1)
      LOGICAL LCORE(1)
      COMMON /DSCORE/ DCORE
      EQUIVALENCE (DCORE(1),RCORE(1),ICORE(1),LCORE(1))
      INTEGER IBUF(11)
      REAL BUF(11)
      EQUIVALENCE (IBUF(1), BUF(1))
      COMMON /PRECIS/EMACH
      EMACH=1.D-14
C
      DATA DSLIST/'VALUE', 'VMIN', 'VMAX'/
      DATA DHLIST/'OBJEXACT', 'NFUNC', 'NGRAD', 'NCON', 'NAC',
     1   'NVC', 'NLBS', 'NUBS', 'CONVRGD', 'NITER', 'OBJAPROX'/
C
      CALL MMBASE (DCORE)
C
C   GET THE DESIGN VARIABLE INFORMATION AND OBTAIN MEMORY
C   TO STORE DESIGN VARIABLE VALUES (BOTH INITIAL AND AFTER
C   REDESIGN), UPPER AND LOWER BOUNDS, AND OBJECTIVE FUNCTION
C   GRADIENTS
C
      CALL DBOPEN (GLBDES, INFO, 'RO', 'NOFLUSH', ISTAT)
      NMAX=NDIM + 1
```

```
C
      CALL MMGETB ('X', 'RSP',5*NMAX, 'SCP', IXO, ISTAT)
      IF (ISTAT.NE.0) CALL UTMCOR(5*NMAX,'RSP','SCP 1')
C
      IX = IXO + NMAX
      IXL = IX + NMAX
      IXU = IXL + NMAX
      IDF = IXU + NMAX
C
C
      CALL RECOND (GLBDES, 'NITER', 'EQ', NITER)
      CALL REENDC
      CALL REPROJ (GLBDES, 1, 'VALUE')
      CALL REGB (GLBDES, RCORE(IXO), NDIM, ISTAT)
      DO 5 I = 1, NDIM
         RCORE(IX+I-1) = RCORE(IXO+I-1)
    5 CONTINUE
C
C
      CALL RECOND (GLBDES, 'NITER', 'EQ', NITER)
      CALL REENDC
      CALL REPROJ (GLBDES, 1, 'VMIN')
      CALL REGB (GLBDES, RCORE(IXL), NDIM, ISTAT)
C
      CALL RECOND (GLBDES, 'NITER', 'EQ', NITER)
      CALL REENDC
      CALL REPROJ (GLBDES, 1, 'VMAX')
      CALL REGB (GLBDES, RCORE(IXU), NDIM, ISTAT)
      CALL DBCLOS (GLBDES)
C
C   GET OBJECTIVE FUNCTION GRADIENTS FROM API INTRINSIC
C
      CALL APWOBJ (ISTOBJ, POSITN)
      CALL GWOBJD (ISTOBJ, POSITN, RCORE(IDF))
C
C   CONVERT TO DOUBLE PRECISION
C
      CALL MMGETB('DX','RDP',4*NMAX,'SCP',IDX,ISTAT)
      IF (ISTAT.NE.0) CALL UTMCOR(4*NMAX,'RDP','SCP 2')
      IDXL = IDX + NMAX
      IDXU = IDXL + NMAX
      IDDF = IDXU + NMAX
      DO 10 I = 1, NDIM
         DCORE(IDX+I-1) = DBLE(RCORE(IX+I-1))
         DCORE(IDXL+I-1) = DBLE(RCORE(IXL+I-1))
         DCORE(IDXU+I-1) = DBLE(RCORE(IXU+I-1))
         DCORE(IDDF+I-1) = DBLE(RCORE(IDF+I-1))
   10 CONTINUE
C
C    GET MEMORY FOR ACTIVE CONSTRAINTS
C
      CALL DBOPEN(A,INFO,'RO','NOFLUSH',ISTAT)
      MG = INFO(2)
      CALL DBCLOS(A)
      MMAX = MG + 1
      MH = 0
C
      CALL MMGETB('CON','RSP',MMAX+MG,'SCP',ICON,ISTAT)
      IF (ISTAT.NE.0) CALL UTMCOR(MMAX+MG,'RSP','SCP 3')
      IACT = ICON + MMAX
C
C    RETRIEVE ACTIVE CONSTRAINT VALUES
C
      CALL DBOPEN(CSTORD,INFO,'RO','NOFLUSH',ISTAT)
      CALL RECOND(CSTORD, 'NITER', 'EQ', NITER)
      CALL REENDC
      CALL REPROJ(CSTORD, 1, 'CVAL')
      CALL REGB(CSTORD, RCORE(ICON), MG, ISTAT)
      CALL DBCLOS (CSTORD)
C
C    RETRIEVE CONSTRAINT ACTIVE FLAGS
C
      CALL DBOPEN(CONST,INFO,'RO','NOFLUSH',ISTAT)
      CALL RECOND (CONST, 'NITER', 'EQ', NITER)
      CALL RESETC ('AND', 'ACTVFLAG', 'EQ', 1)
      CALL REENDC
      CALL REPROJ(CONST, 1, 'ACTVFLAG')
      CALL REGB(CONST, ICORE(IACT), MG, ISTAT)
      CALL DBCLOS(CONST)
C
      CALL MMGETB('G','RDP',MMAX,'SCP',IG,ISTAT)
      IF (ISTAT.NE.0) CALL UTMCOR(MMAX,'RDP','SCP 4')
      CALL MMGETB('GACT','RSP',MG,'SCP',IACTIVE,ISTAT)
      IF (ISTAT.NE.0) CALL UTMCOR(MG,'RSP','SCP 5')
C
C   CONVERT CONSTRAINT VALUES TO DOUBLE PRECISION AND
C   ACTIVE FLAG TO LOGICAL
C
      DO 20 I = 1, MG
         DCORE(IG+I-1) = -1.0D0*DBLE(RCORE(ICON+I-1))
         LCORE(IACTIVE+I-1) = .FALSE.
         IF (ICORE(IACT+I-1).EQ.1)
     +                LCORE(IACTIVE+I-1) = .TRUE.
   20 CONTINUE
C
C  GET MEMORY FOR CONSTRAINT GRADIENTS
C
      CALL MMGETB('TEMP','RDP',NDIM,'SCP',ITEMP,ISTAT)
      IF (ISTAT.NE.0) CALL UTMCOR(NDIM,'RDP','SCP 6')
      CALL MMGETB('DG','RDP',MMAX*NMAX,'SCP',IDG,ISTAT)
      IF (ISTAT.NE.0) CALL UTMCOR(MMAX*NMAX,'RDP','SCP 7')
C
C  RETRIEVE MATRIX OF ACTIVE CONSTRAINT GRADIENTS
C
      CALL DBOPEN('AMAT',INFO,'RO','NOFLUSH',ISTAT)
      DO 30 I = 1, MG
         CALL MXUNP('AMAT',DCORE(ITEMP),1,NDIM)
         DO 40 J = 1,NDIM
            DCORE(IDG+MMAX*(J-1)+(I-1))=-DCORE(ITEMP+J-1)
   40    CONTINUE
   30 CONTINUE
      CALL DBCLOS('AMAT')
C
C  CALCULATE OBJECTIVE FUNCTION VALUE
C
      FO = 0.D0
      DO 50 I = 1, NDIM
         FO = FO + DCORE(IDX+I-1)*DCORE(IDDF+I-1)
   50 CONTINUE
C
C  ALLOCATE AND INITIALIZE MEMORY REQUIRED BY SCP
C
      MNN = MMAX + NMAX + NMAX
      RDIMN = MG*MG + NDIM*MG + 24*NDIM + 14*MG + 9
      LDIMN = 4*NDIM + 3*MG + 2
      IDIMN = 2*MG + 6
C
      CALL MMGETB('U','RDP',MNN,'SCP',IU,ISTAT)
      IF (ISTAT.NE.0) CALL UTMCOR(MNN,'RDP','SCP 8')
      IUL = IU + MMAX
      IUU = IUL + NMAX
C
      CALL MMGETB('WAS','RDP',MMAX+RDIMN,'SCP',WA,ISTAT)
      IF (ISTAT.NE.0) CALL UTMCOR(MMAX,'RDP','SCP 9')
      RDIM = WA + MMAX
```

```
C
      CALL MMGETB('ISCP','RSP',IDIMN+LDIMN,
     +                         'SCP',IDIM,ISTAT)
      IF (ISTAT.NE.0) CALL UTMCOR(2*MG+6+LDIMN,
     +                            'RSP','SCP 10')
      LDIM = IDIM + IDIMN
C
C  DEFINE SCP PARAMETERS
C
      ACC=1.D-4
      ITMAX=1
      MAXINT=10
      ITDUAL=1000
      IPRINT=2
      NOUT=7
      ACTRES=1.D5
      T1=0.1D0
      T2=1.5D0
      GAMMA=0.5D0
      METHOD='S'
      STRAT='P'
      MODE=2
      IERR=0
C
C  INITIALIZE COUNTERS
C
      NRF = 0
      NRG = 0
C
C  CALL THE SCP OPTIMIZER, GETTING NEW DESIGN POINT
C
555   CALL ASCP(NDIM,MG,MH,MMAX,NMAX,DCORE(IDX),DCORE(IDXL),
     +          DCORE(IDXU),FO,DCORE(IG),DCORE(IDDF),
     +          DCORE(IDG),DCORE(IU),DCORE(IUL),DCORE(IUU),
     +          ACC,ITMAX,NRF,NRG,MAXINT,ITDUAL,IPRINT,NOUT,
     +          ACTRES,T1,T2,GAMMA,DCORE(WA),DCORE(RDIM),
     +          RDIMN,ICORE(IDIM),IDIMN,LCORE(LDIM),LDIMN,
     +          LCORE(IACTIVE),METHOD,STRAT,MODE,IERR)
C
      IF (IERR .EQ. -2) THEN
C
C  GRADIENTS REQUESTED BY SCP
C
         GOTO 555
      ELSE IF (IERR .EQ. -1) THEN
```

```
C
C  FUNCTION VALUES REQUESTED BY SCP
C
         GOTO 555
      END IF
C
C  SET GLOBAL CONVERGENCE PARAMETERS
C
      CTL = -0.0001
      CTLMIN = 0.0005
C
C  CALCULATE NEW, APPROXIMATE OBJECTIVE FUNCTION VALUE
C
      F = 0.D0
      DO 55 I = 1, NDIM
         F = F + DCORE(IDX+I-1)*DCORE(IDDF+I-1)
   55 CONTINUE
C
C  CONVERT NEW DESIGN VARIABLES BACK TO SINGLE PRECISION
C
      DO 60 I = 1, NDIM
         RCORE(IX+I-1) = (DCORE(IDX+I-1))
   60 CONTINUE
C
C   UPDATE GLBDES RELATION
C
C   BRING IN THE LIST OF DESIGN VARIABLE ID'S
C
      CALL DBOPEN (GLBDES,INFO,'R/W','NOFLUSH',ISTAT)
      CALL REPROJ (GLBDES,1,'DVID')
      CALL RECOND (GLBDES,'NITER','EQ',NITER)
      CALL REENDC
      CALL MMGETB ('DVID','RSP',NDIM,'SCP',IDV,ISTAT)
      IF (ISTAT.NE.0) CALL UTMCOR(NDIM,'RSP','SCP 11')
      CALL REGB (GLBDES,ICORE(IDV),NDIM,ISTAT)
      CALL DBCLOS (GLBDES)
C
C   CALL THE DVUPDT UTILITY TO UPDATE THE RELATION
C
      CALL DVUPDT(NITER,NDIM,ICORE(IDV),RCORE(IX),GLBDES,
     +       'SCP',ICORE,ICORE)
C
C   WRITE A NEW ENTRY TO THE DESHIST RELATION
C
C
```

```
      CALL DBOPEN (DESHIST,INFO,'WO','NOFLUSH',ISTAT)
      CALL REPROJ (DESHIST,11,DHLIST)
      IF (INFO(3) .EQ. 0) THEN
         BUF(1) = FO
         CALL UTZERS(IBUF(2), 8, 0)
         IBUF(10) = NITER
         CALL READD (HIST, BUF)
      END IF
      IBUF(10) = NITER + 1
      BUF(1) = FO
      BUF(11) = F
      IBUF(2) = NRF
      IBUF(3) = NRG
      IBUF(4) = MG
C
C   DETERMINE NUMBER OF ACTIVE AND VIOLATED CONSTRAINTS
C
      NAC = 0
      NVC = 0
      DO 70 I = 0, MG -1
         IF (RCORE(ICON+I) .GE. CTL) THEN
            NAC = NAC + 1
            IF (RCORE(ICON+I) .GT. CTLMIN) NVC = NVC + 1
         END IF
   70 CONTINUE
      IBUF(5) = NAC
      IBUF(6) = NVC
C
C   DETERMINE NUMBER OF ACTIVE SIDE CONSTRAINTS
C
      NLBS = 0
      NUBS = 0
      DO 80 I = 0, NDIM - 1
         IF (ABS(RCORE(IX+I)-RCORE(IXL+I)) .LE. CTLMIN)
     +      NLBS = NLBS + 1
         IF (ABS(RCORE(IXU+I)-RCORE(IX+I)) .LE. CTLMIN)
     +      NUBS = NUBS + 1
   80 CONTINUE
      IBUF(7) = NLBS
      IBUF(8) = NUBS
C
C  DETERMINE STATUS OF APPROXIMATE PROBLEM CONVERGENCE
C
      CALL ACNVRG (NITER, APPCNVRG, CNVRGLIM, 'MP', NDIM,
     +             ICORE(IDV), RCORE(IXO), RCORE(IX), FO, F)
```

```
      IF (APPCNVRG) THEN
        IBUF(9) = 1
      ELSE
        IBUF(9) = 0
      END IF
      CALL READD (DESHIST,BUF)
      CALL DBCLOS (DESHIST)
      PRINT *, 'ITERATION ', NITER
      PRINT *, 'FUNCTION CALLS, NRF = ', NRF
      PRINT *, 'GRADIENT CALLS, NRG = ', NRG
      CALL MMFREG ('SCP')
C
      RETURN
      END
***********************************************************************
C
C   THESE SUBROUTINES ARE USED FOR EXPLICITLY DEFINED FUNCTIONS.
C   THEY ARE EMPTY FOR STRUCTURAL, FINITE-ELEMENT APPLICATIONS.
C
***********************************************************************
      SUBROUTINE NLFUNC (M,ME,MMAX,N,F,G,X,ACTIVE)
      RETURN
      END
***********************************************************************
      SUBROUTINE NLGRAD (M,ME,MMAX,N,F,G,DF,DG,X,
     +                   ACTIVE,WA)
      RETURN
      END
```

Bibliography

1. Abramson, M.A. *Application of Sequential Quadratic Programming to Large-Scale Structural Design Problems.* Thesis, Graduate School of Engineering, Air Force Institute of Technology, 1994.

2. Abramson, M.A. and J.W. Chrissis. "Sequential Quadratic Programming and the ASTROS Structural Optimization System," *Structural Optimization, 15*:24-32 (1998).

3. Arora, J.S. *Introduction to Optimum Design*, McGraw-Hill, 1989.

4. Barthelemy, J.-F.M. and R.T. Haftka. "Function Approximations," *Structural Optimization: Status and Promise* edited by M.P. Kamat, chapter 4, 51-70, AIAA, 1993.

5. Beckers, M. and C. Fleury. "A Primal-Dual Approach in Truss Topology Optimization," *Computers & Structures, 64*(1-4):77-88 (1997).

6. Ben-Tal, A. and G. Roth. "A Truncated Log Barrier Algorithm for Large-Scale Convex Programming and MinMax Problems: Implementation and Computational Results," *Optimization Methods and Software, 6*:283-312 (1996).

7. Bletzinger, K.-U. "Extended Method of Moving Asymptotes Based on Second-Order Information," *Structural Optimization, 5*:175-183 (1993).

8. Boyd, S. and L. Vandenberghe. "CRCD Program: Convex optimization for Engineering Analysis and Design," Proceedings of the American Control Conference, 2: 1069-71 (1995).

9. Canfield, R.A. and V.B. Venkayya. "Implementation of Generalized Optimality Criteria in a Multidisciplinary Environment," *Journal of Aircraft, 27*:1037-42 (1990).

10. Duysinx, P. and C. Fleury. "Optimization Software: View from Europe," *Structural Optimization: Status and Promise* edited by M.P. Kamat, chapter 28, 807-849, AIAA, 1993.

11. Fleury, C. "Structural Weight Optimization by Dual Methods of Convex Programming," *International Journal for Numerical Methods in Engineering, 14*:1761-83 (1979).

12. Fleury, C. and V. Braibant. "Structural Optimization: A new Dual Method Using Mixed Variables," *International Journal for Numerical Methods in Engineering, 23*:409-428 (1986).

13. Fleury, C. "Efficient Approximation Concepts Using Second Order Information," *International Journal for Numerical Methods in Engineering, 28*:2041-58 (1989).

14. Fleury, C. "First and Second Order Convex Approximation Strategies in Structural Optimization," *Structural Optimization, 1*:3-10 (1989).

15. Fleury, C. "CONLIN: An Efficient Dual Optimizer Based on Convex Approximation Concepts," *Structural Optimization, 1*:81-89 (1989).

16. Fleury, C. "Sequential Convex Programming for Structural Optimization Problems," *Optimization of Large Structural Systems, 1* of *NATO ASI* edited by G.I.N. Rozvany, 531-553, Kluwer Academic Publishers, 1993.

17. Fleury, C. "Dual Methods for Convex Separable Problems," *Optimization of Large Structural Systems, 1* of *NATO ASI* edited by G.I.N. Rozvany, 509-530, Kluwer Academic Publishers, 1993.

18. Fleury, C. "Mathematical Programming Methods for Constrained Optimization: Dual Methods," *Structural Optimization: Status and Promise* edited by M.P. Kamat, chapter 7, 123-150, AIAA, 1993.

19. Fleury, C. "Recent Developments in Structural Optimization Methods," *Structural Optimization: Status and Promise* edited by M.P. Kamat, chapter 9, 183-208, AIAA, 1993.

20. Jarre, F. and M. Saunders. "A Practical Interior-Point Method for Convex Programming," *SIAM Journal on Optimization, 5*(1):149-171 (1995).

21. Johnson, E.W. "Tools for Structural Optimization," *Structural Optimization: Status and Promise* edited by M.P. Kamat, chapter 29, 851-863, AIAA, 1993.

22. Jonsson, Ö and T. Larsson. "A note On Step-Size Restrictions in Approximation Procedures for Structural Optimization," *Computers & Structures, 37*(3):259-263 (1990).

23. Kirsch, U. *Optimum Structural Design*, McGraw-Hill, 1989.

24. Larsson, T. and M. Rönnqvist. "A Method for Structural Optimization Which Combines Second-Order Approximations and Dual Techniques," *Structural Optimization, 5*:225-232 (1993).

25. Ma, Z.-D. and N. Kikuchi. "A New Method of Sequential Approximate Optimization for Structural Optimization Problems," *Engineering Optimization, 25*:231-253 (1995).

26. Mahmoud, K.G. "An Efficient Approach to Structural Optimization," *Computers & Structures, 64*(1-4):97-112 (1997).

27. Neill, D.J. and R.A. Canfield. "ASTROS—A Multidisciplinary Automated Structural Design Tool," AIAA Paper 87-0713, 44-53, April 1987.

28. Neill, D.J. and D.L. Herendeen. *ASTROS Enhancements, Volume I-ASTROS User's Manual.* Technical Report WL-TR-96-3004, Wright Laboratory, May, 1995.

29. Neill, D.J. and D.L. Herendeen. *ASTROS Enhancements, Volume II-ASTROS Programmer's Manual.* Technical Report WL-TR-96-3005, Wright Laboratory, May, 1995.

30. Neill, D.J., D.L. Herendeen, and V.B. Venkayya. *ASTROS Enhancements, Volume III-ASTROS Theoretical Manual.* Technical Report WL-TR-96-3006, Wright Laboratory, May, 1995.

31. Nesterov, Y. and A. Nemirovskii. *Interior Point Polynomial Methods in Convex Programming*, Philadelphia, SIAM, 1994.

32. Rao, S.S. *Engineering Optimization: Theory and Practice*, John Wiley and Sons, 1996.

33. Reklaitis, G.V., A. Ravindran, and K.M. Ragsdell. *Engineering Optimization: Methods and Applications*, John Wiley and Sons, 1983.

34. Schittkowski, K., C. Zillober and R. Zotemantel. "Numerical Comparison of Nonlinear Programming Algorithms for Structural Optimization," *Structural Optimization, 7*:1-19 (1994).

35. Schittkowski, K. and C. Zillober. "Sequential Convex Programming Methods," in *Stochastic Programming*, edited by K. Marti and P. Kall, Lecture Notes in Economics and Mathematical Systems, vol. 423, Springer, 1995.

36. Schmit Jr., L.A. and B. Farshi. "Some Approximation Concepts for Structural Synthesis," *AIAA Journal, 12*(5):692-699 (May 1974).

37. Schmit Jr., L.A. "Structural Optimization—Some Key Ideas and Insights", *New Directions in Optimum Structural Design*, edited by E. Atrek, R.H. Gallagher, K.M. Ragsdell and O.C. Zienkiewicz, chapter 1, 1-45, Wiley, 1984.

38. Svanberg, K. "The Method of Moving Asymptotes—a New Method for Structural Optimization," *International Journal for Numerical Methods in Engineering, 24*:359-373 (1987).

39. Svanberg, K. "Some Second Order Methods for Structural Optimization," *Optimization of Large Structural Systems, 1* of *NATO ASI* edited by G.I.N. Rozvany, 567-578, Kluwer Academic Publishers, 1993.

40. Svanberg, K. "The Method of Moving Asymptotes (MMA) with Some Extensions," *Optimization of Large Structural Systems, 1* of *NATO ASI* edited by G.I.N. Rozvany, 555-566, Kluwer Academic Publishers, 1993.

41. Thanedar, P.B., J.Arora, C.H.Tseng, O.K.Lim, and G.J.Bark "Performance of Some SQP Algorithms on Structural Design Problems," *International of Numerical Methods in Engineering, 23*(3):2187-2203 (1986).

42. Vanderplaatz, G.N. "An Efficient Feasible Directions Algorithm for Design Synthesis," *AIAA Journal, 22*(11):1633-1640 (November 1984).

43. Vanderplaatz, G.N. *Numerical Optimization Techniques for Engineering Design: With Applications*, McGraw-Hill, 1984.

44. Vanderplaatz, G.N. "Structural Optimization," *Computational Nonlinear Mechanics in Aerospace Engineering* edited by S.T. Atluri, chapter 14, 507-529, AIAA, 1992.

45. Vanderplaatz, G.N. "Mathematical Programming Methods for Constrained Optimization: Primal Methods", *Structural Optimization: Status and Promise* edited by M.P. Kamat, chapter 3, 29-49, AIAA, 1993.

46. Venkayya, V.B. "Optimality Criteria: A Basis for Multidisciplinary Design Optimization," *Computational Mechanics, 5*(1):1-21 (1989).

47. Venkayya, V.B., V.A. Tischler, and S.M. Pitrof. "Benchmarking in Structural Optimization," *Proceedings of the 4th AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization*, Sep. 21-23, 1992, Cleveland, OH, AIAA Paper AIAA-92-4784.

48. Venkayya, V.B. "Introduction: Historical Perspective and Future Directions", *Structural Optimization: Status and Promise* edited by M.P. Kamat, chapter 1, 1-10, AIAA, 1993.

49. Venkayya, V.B. "Generalized Optimality Criteria Method", *Structural Optimization: Status and Promise* edited by M.P. Kamat, chapter 8, 151-182, AIAA, 1993.

50. Zhang, W. and C. Fleury. "A Modification of Convex Approximation Methods for Structural Optimization," *Computers & Structures, 64*(1-4):89-95 (1997).

51. Zillober, C. *SCP - An Implementation of a Sequential Convex Programming Algorithm for Nonlinear Programming*, DFG-Report No. 470, Schwerpunkt Anwendungsbezogene Optimierung and Steuerung, 1993.

52. Zillober, C. unpublished SCP algorithm FORTRAN source code.

53. Zillober, C. "A Globally Convergent Version of the Method of Moving Asymptotes," *Structural Optimization, 6*:166-174 (1993).

Vita

Capt. Todd A. Sriver was born on 22 November 1967 in Plymouth, Indiana. He graduated from Riley High School in South Bend, Indiana in 1986 and then attended Purdue University in West Lafayette, Indiana. He graduated with a Bachelor of Science degree in Aeronautical and Astronautical Engineering in December 1990. He received a reserve commission in the USAF on 22 September 1993 upon graduation from Officer Training School.

Captain Sriver's first assignment was with Detachment 2, Space and Missiles Systems Center at Onizuka Air Station, California. Assigned to the Engineering Directorate, he was responsible for the analysis and planning of over 100 new R&D satellite control requirements per year. He was also the project officer for the integration and testing of the Interim RDT&E Support Complex (IRSC), a prototype satellite command and control system. In August 1996, he entered the Graduate School of Engineering, Air Force Institute of Technology as a student in Operations Research. Upon graduation, Captain Sriver is assigned to the 33rd Flight Test Squadron, Air Mobility Command, McGuire AFB, New Jersey.

Captain Sriver is a member of the Tau Beta Pi and Omega Rho national honor societies.

Captain Sriver and his wife, the former Stephanie Jo Shroyer of Mishawaka, Indiana, have two children: Tyler and Kayla.

Permanent address: 19880 Yoder Street
South Bend, Indiana 46614

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 1998	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE THE APPLICATION OF SEQUENTIAL CONVEX PROGRAMMING TO LARGE-SCALE STRUCTURAL OPTIMIZATION PROBLEMS	**5. FUNDING NUMBERS**
6. AUTHOR(S) Todd Allen Sriver, Captain, USAF	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology 2950 P Street Wright-Patterson AFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GOR/ENS/98M-24
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Dr. Vipperla Venkayya 2130 8th Street, Suite 1 AFRL/VASD Wright-Patterson AFB OH 45433-7542	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES

Advisor: Dr. James W. Chrissis, (937) 255-6565 ext. 4338, jchrissi@afit.af.mil

12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for public release; distribution unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 words)*

Structural design problems are often modeled using finite element methods. Such models are often characterized by constraint functions that are not explicitly defined in terms of the design variables. These functions are typically evaluated through numerical finite element analysis (FEA). Optimizing large-scale structural design models requires computationally expensive FEAs to obtain function and gradient values. An optimization approach which uses the SCP sequential convex programming algorithm of Zillober, integrated as the optimizer in the Automated Structural Optimization System (ASTROS), is tested. The traditional approach forms an explicitly defined approximate subproblem at each design iteration that is solved using the method of modified feasible directions. In an alternative approach, the SCP subroutine is called to formulate and solve the approximate subproblem. The SCP method is an implementation of the Method of Moving Asymptotes algorithm with five different asymptote determination strategies. This study investigates the effect of different asymptote determination strategies and constraint retention strategies on computational efficiency. The approach is tested on three large-scale structural design models, including one with constraints from multiple disciplines. Results and comparisons to the traditional approach are given. The largest of the three models, which had 1527 design variables and 6124 constraints, was solved to optimality with ASTROS for the first time using a mathematical programming method. The structural weight of the resulting design is 9% lower than the previously recorded minimum weight.

14. SUBJECT TERMS Optimization; Structural Optimization; Nonlinear Programming; Sequential Convex Programming	**15. NUMBER OF PAGES** 90
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UL